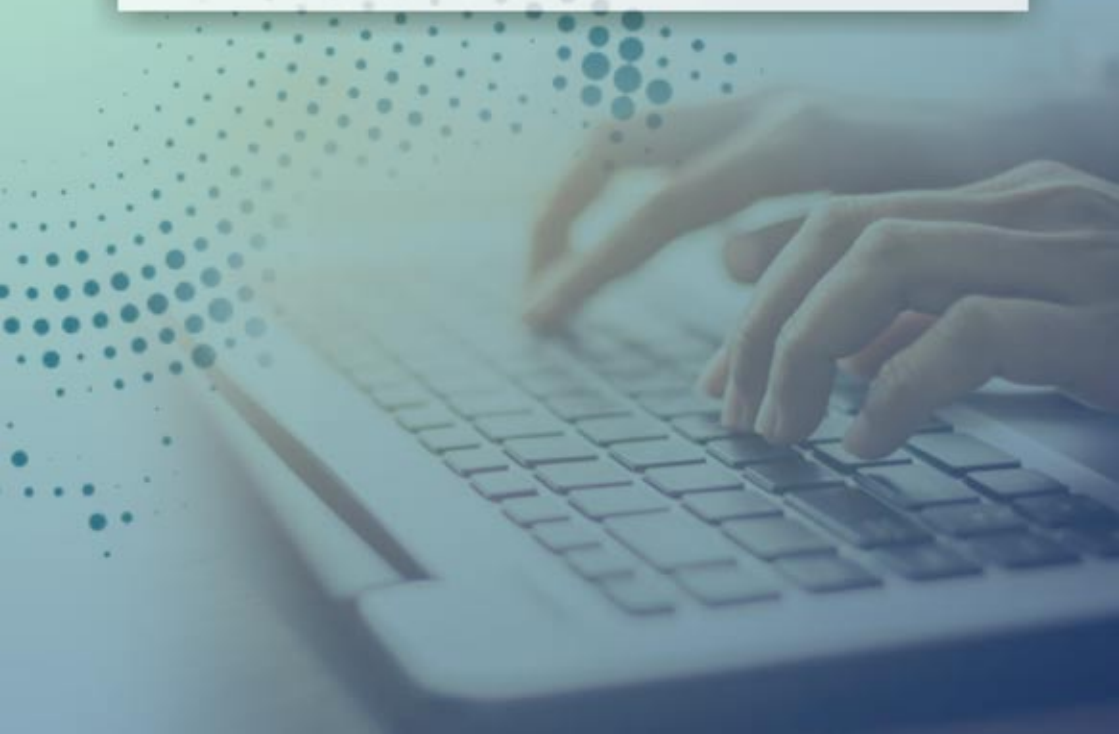
Е. Б. ГРИБАНОВА
Р. С. ГЕРАСИМОВ

АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ МЕТОДОВ РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ



Е. Б. Грибанова
Р. С. Герасимов

АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ МЕТОДОВ РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ

Монография

Чебоксары
Издательский дом «Среда»
2026

УДК 004.85
ББК 32.813
Г82

Рецензенты:
д-р техн. наук, профессор
ФГАОУ ВО «Томский государственный университет систем управления и радиоэлектроники»
Артур Александрович Мицель;
канд. физ-мат. наук, доцент
ФГБОУ ВО «Чувашский государственный университет им. И.Н. Ульянова»
Евгения Арифжановна Микишанина

Грибанова Е. Б.
Г82 **Алгоритмы машинного обучения на основе методов решения обратных задач :** монография / Е. Б. Грибанова, Р. С. Герасимов. – Чебоксары : Среда, 2026. – 100 с.

ISBN 978-5-908172-30-1

Монография посвящена проблеме повышения эффективности алгоритмов машинного обучения на основе математического аппарата решения некорректно поставленных обратных задач. Целью работы является разработка и тестирование новых оптимизационных алгоритмов, предназначенных для обучения регрессионных и нейросетевых моделей, а также кластеризации данных.

В исследовании выполнено переформулирование задач оптимизации в машинном обучении в виде обратных задач. Показано, что применение адаптивного стохастического поиска и L1-регуляризации позволяет в ряде случаев не только повысить точность предсказаний, но и создавать компактные, сильно разреженные решения при минимальной настройке гиперпараметров (подход «из коробки»). Предложенные гибридные алгоритмы позволили эффективно отсеять незначимые признаки без значительной потери прогностической точности, что сделало логику работы моделей прозрачной. Апробация алгоритмов выполнена на верифицированных датасетах репозиториев KEEL и UCI, реальных данных из разных сфер (экономика, метеорология и т.д.), а также синтетических данных.

Монография предназначена для математиков, интересующихся применением методов оптимизации в сфере Data Science, а также для специалистов по машинному обучению, ищущих новые подходы для повышения качества, устойчивости и интерпретируемости разрабатываемых моделей.

Исследование выполнено за счет гранта Российского научного фонда № 25-21-00123, https://rscf.ru/project/25-21-00123/

ISBN 978-5-908172-30-1
DOI 10.31483/a-10940

© Грибанова Е.Б.,
Герасимов Р. С., 2026
© ИД «Среда», оформление, 2026

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5
1. Методы решения обратных задач 7
1.1. Обратные задачи: понятие и виды 7
1.2. Решение обратных задач с помощью обратных вычислений..... 9
1.2.1. Стохастический метод решения обратных задач с ограничениями 12
1.3. Решение обратной задачи с использованием регуляризации... 15
1.3.1. Метод решения обратной задачи с L1-регуляризацией 17
2. Алгоритмы машинного обучения на основе стохастического метода решения обратных задач 20
2.1. Глобальная оптимизация функции..... 20
2.1.1. Стохастический алгоритм глобальной оптимизации 21
2.1.2. Результаты экспериментов..... 23
2.2. Обучение полносвязных нейронных сетей 29
2.2.1. Алгоритмы обучения нейронной сети 30
2.2.2. Результаты экспериментов..... 35
2.3. Кластеризация данных 40
2.3.1. Алгоритм кластеризации..... 40
2.3.2. Результаты экспериментов..... 42
3. Алгоритмы машинного обучения на основе метода решения обратных задач с L1-регуляризацией 51
3.1. Обучение LASSO-регрессии..... 51
3.1.1. Алгоритм обучения..... 52
3.1.2. Гибридный алгоритм обучения 55
3.1.3. Результаты экспериментов..... 56
3.2. Обучение полносвязной нейронной сети 66

3.2.1. Алгоритм обучения 66
3.2.2. Результаты экспериментов 67
3.3. Гибридное обучение нейронной сети с отбором признаков 73
3.3.1. Алгоритм обучения 74
3.3.2. Результаты экспериментов 76
3.4. Гибридное обучение разреженной нейронной сети без отбора признаков 79
3.4.1. Алгоритм обучения 80
3.4.2. Результаты экспериментов 82
ЗАКЛЮЧЕНИЕ 90
ЛИТЕРАТУРА 91

ВВЕДЕНИЕ

В последние десятилетия машинное обучение стало одним из важнейших инструментов анализа данных, извлечения закономерностей и моделирования сложных процессов. Бурное развитие технологий искусственного интеллекта обусловлено беспрецедентным ростом объемов доступных данных и увеличением вычислительных мощностей. Однако по мере усложнения моделей и роста требований к их надежности, на первый план выходят проблемы эффективности оптимизации, устойчивости алгоритмов и, что особенно важно, интерпретируемости получаемых результатов. В этом контексте интеграция концепций машинного обучения с фундаментальными математическими подходами становится перспективным направлением исследований.

Особое место в этом синтезе занимают методы решения обратных задач. Исторически теория обратных задач развивалась для восстановления физических параметров систем по косвенным наблюдениям. Вместе с тем многие классические и современные задачи машинного обучения по своей математической природе могут быть переформулированы в виде последовательности некорректно поставленных обратных задач. В них требуется найти неизвестные параметры модели (причины) по значению функции потерь (следствиям). Такое концептуальное родство позволяет применять математический аппарат регуляризации и итеративной оптимизации, разработанный для обратных задач, к задачам машинного обучения.

Цель исследования заключается в разработке оптимизационных алгоритмов, которые позволяют эффективно решать задачи машинного обучения, такие как обучение регрессионных и нейросетевых моделей, а также кластеризация данных, используя методы решения обратных задач. В работе подробно рассматриваются как сами пошаговые алгоритмы, так и их тестирование на верифицированных датасетах.

Как показывают представленные в монографии результаты, применение методов обратных задач к задачам машинного обучения обеспечили следующие преимущества. В ряде задач это способствовало повышению точности предсказаний и ускорению сходимости алгоритмов обучения за счет использования адаптивного

стохастического поиска и методов регуляризации. Это позволило создать более компактные модели, сопоставимые по точности с более сложными аналогами. При этом в ряде экспериментов настройка гиперпараметров не применялась вовсе или была минимальной, чтобы оценить алгоритмы в качестве «коробочного» решения. Кроме того, методы решения обратных задач позволили сформировать сильно разреженные решения. Это качество критически важно для отбора значимых признаков и создания интерпретируемых моделей, что особенно актуально в медицине, финансах и критических инженерных системах, где «цена ошибки» алгоритма крайне высока.

Предложенные алгоритмы и полученные результаты могут найти применение при разработке широкого спектра приложений в различных сферах науки, экономики и производства. Монография будет полезна как математикам, интересующимся применением оптимизационных методов в Data Science, так и специалистам по машинному обучению, ищущим новые инструменты для повышения качества и интерпретируемости своих моделей.

1. МЕТОДЫ РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ

1.1. Обратные задачи: понятие и виды

Под прямыми и обратными задачами понимают два типа постановки проблемы, различающиеся направлением логического вывода: от причины к следствию или от наблюдаемого следствия к неизвестной причине [1–2]:

- Прямая задача – это определение следствий (выходных данных, поведения системы) по известным причинам (входным данным, параметрам, уравнениям модели).
- Обратная задача – это восстановление неизвестных причин (параметров модели, начальных/граничных условий, структуры объекта) по наблюдаемым следствиям (измерениям, результатам).

Обратные задачи считаются более сложными по сравнению с прямыми из-за их некорректности. Определение корректности задачи было сформулировано Ж. Адамаром, в частности, был обозначен такой признак как наличие единственного решения задачи на некотором множестве, которое должно непрерывно зависеть от входных данных.

Значительный вклад в исследование обратных задач принадлежит А. Н. Тихонову [3,4], который показал, что определение дополнительных условий на решение позволяет получить устойчивую задачу. Он предложил способ регуляризации некорректной задачи, т.е. сведение исходной задачи решения некоторого операторного уравнения к проблеме поиска минимума некоторого функционала.

Одинцовым Б.Е. были исследованы обратные задачи, которые в отличие от содержания задач, рассмотренных Тихоновым А.Н. и другими авторами, характеризуются целью управления и наличием прямой функции, отражающей зависимость следствия от причины. Такие задачи являются трудоемкими и капризными [5], что выражается в непредсказуемости поведения функции и сложностью определения диапазонов входных данных, при которых задача имеет смысл и решение. Решение данного класса задач может быть точечными, т.е. базироваться на некоторой точке области решения. В качестве метода решения Одинцовым Б.Е. был предложен аппарат обратных вычислений, применение которого требует указания лицом, принимающим решения, прямой зависимости и целевой установки. Таким образом, получаются точечные значения приростов аргументов прямой функции (аргументы определяются только для одной заданной точки функции) на основании её задаваемого значения и дополнительной экспертной информации, в качестве которой выступает направление изменения показателей и коэффициенты относительной важности. Такая задача будет в данной работе в дальнейшем именоваться одноточечной обратной задачей. При этом прямые вычисления здесь позволяют получить начальное приближение решения задачи.

На рисунке 1.1 показана такая одноточечная обратная задача формирования показателя и заключается в определении набора исходных величин x или их изменений, обеспечивающих заданное значение результирующего показателя y^*.

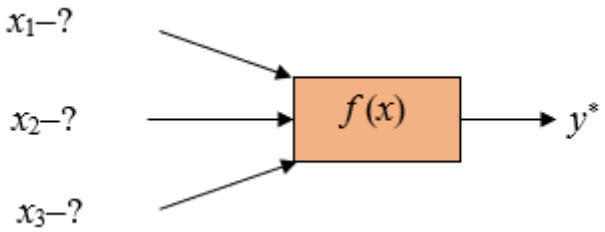


Рис. 1.1. Обратная задача: x – входные параметры;
y* – заданное значение выходной величины

1.2. Решение обратных задач с помощью обратных вычислений

Обратную задачу (рисунок 1.1) при многоуровневых показателях можно представить в виде древовидной структуры (рисунок 1.2): заданы исходные значения показателей подцели первого (x_1, x_2, x_3) и второго (x_{11}, x_{21}, x_{22}) уровня, необходимо определить их изменения (Δx_1, Δx_2, Δx_3, Δx_{11}, Δx_{21}, Δx_{22}) таким образом, чтобы целевой показатель был равен y^*.

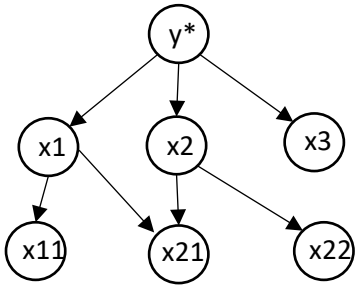


Рис. 1.2. Дерево решения

Для решения поставленной задачи необходимо выполнить два шага:

решение отдельных подзадач;

решение общей задачи формирования результирующего показателя дерева цели.

Под отдельной подзадачей понимается решение обратной задачи по определению значений характеристик, формирующих показатель более высокого уровня. Так, на рисунке 1.2 решение общей задачи будет включать такие подзадачи как определение значений Δx1, Δx2, Δx3 для формирования y*; вычисление Δx11, Δx21 для формирования x1 + Δx1; расчёт Δx21, Δx22 для формирования x2 + Δx2.

Для решения задачи двух аргументов с использованием коэффициентов относительной важности определяется следующая информация:

– *начальные значения аргументов* (x_1, x_2) *и функции* (y);

– *новое значение функции* ($y^* = y \pm \Delta y$);

– *коэффициенты относительной важности аргументов* (β_1, β_2), *определяющие пропорции изменения аргументов;*

– *направление изменений аргументов (+,–), которые устанавливают, будет ли значение аргумента уменьшено либо увеличено.*

На рисунке 1.3 приведено дерево целей. На нём кроме исходных значений показателей x_1 и x_2 отражено также изменение показателя самого верхнего уровня, а также коэффициенты относительной важности β и направления изменения показателей (+, -).

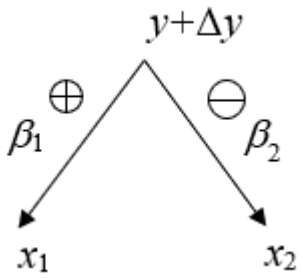


Рис. 1.3. Дерево целей

Для определения изменений аргументов формируется система уравнений:

$$\begin{cases} y \pm \Delta y = f(x_1 \pm \Delta x_1(\beta_1), x_2 \pm \Delta x_2(\beta_2)); \\ \dfrac{\Delta x_1}{\Delta x_2} = \dfrac{\beta_1}{\beta_2}; \\ \beta_1 + \beta_2 = 1. \end{cases}$$

Решением системы будут величины Δx_1 *и* Δx_2*, обеспечивающее значение результирующего показателя, равное* $y \pm \Delta y$.

Уравнения системы отражают соответствие изменений аргументов коэффициентам относительной важности и направлению изменения показателей и равенство результирующего показателя заданному значению.

1.2.1. Стохастический метод решения обратных задач с ограничениями

Вместе со стандартными характеристиками (начальные значение x, y; заданное/искомое значение y^*, коэффициенты относительной важности β) определим следующие дополнительные элементы дерева целей [6–7] (рисунок 1.4):

– минимальное l и максимальное h значения, которые может принимать данный показатель;

– индикатор u, характеризующий возможность использования данного элемента, и принимающий два значения: 1 (использование возможно) и 0 (использованием невозможно).

Значение индикатора u становится равным нулю в случае, если изменение аргумента не может быть выполнено из-за существующего ограничения или отсутствия положительного изменения функции.

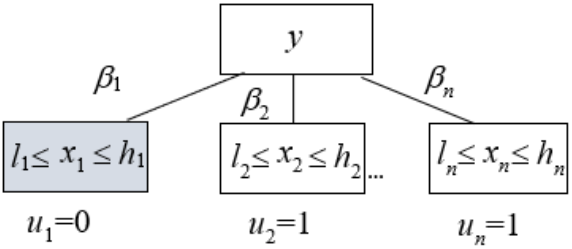


Рис. 1.4. Дерево целей с учётом ограничений

Суть предложенного метода заключается в последовательном изменении результирующего показателя и случайном выборе аргумента для корректировки с помощью моделирования полной группы несовместных событий. При этом коэффициенты важности рассматриваются в качестве вероятности.

Для его применения устанавливается шаг Δy^**, на который будет происходить изменение результирующего показателя (s – номер итерации, y – начальное значение результирующего показателя). Выполнение алгоритма включает следующие шаги.*

Шаг 1. Установить новое значение результирующего показателя

$$y_s = y_s + \Delta y^*.$$

Шаг 2. С помощью алгоритма моделирования полной группы несовместных событий выбрать аргумент k в соответствии с коэффициентами важности $\beta_1, \beta_2, ..., \beta_n$. *Для этого для аргументов с и=1 выполняется расчет значений вероятностей по формуле:*

$$p_i^* = \frac{\beta_i}{\sum_{j=1}^{q} \beta_j}, \; i = 1..q$$

где j *– номер аргумента, для которой значение индикатора и равно 1;*
q – число аргументов, для которых значение индикатора и равно 1.

Шаг 3. Определяется значение $x^*_{s,k}$ *выбранной на предыдущем шаге вершины k, при фиксированных значениях остальных величин для получения заданного* y_s. *Другими словами, происходит решение уравнения* $f(x^*_{s,k}) = y_s$.

Выполняется проверка соответствия полученного значения ограничению $l_k \le x^*_{s,k} \le h_k$. *Если условие выполняется, то* $x_{s,k} = x^*_{s,k}$, *и всем вершинам, не являющимся константой, присваивается индикатор, равный единице, а иначе –* $u_k = 0$, *переход к шагу 1.*

Шаг 4. Осуществляется проверка условия: $y^* < y_s$ *(достижение заданного значения результирующего показателя). Если условие выполняется, происходит завершение работы алгоритма, иначе – переход к шагу 1.*

Полученные значения x_s *будут решением задачи.*

В алгоритме представлена ситуация, когда целевое значение результирующего показателя больше его исходной величины. Если задача заключается в уменьшении его значения, то величина Δy^* *будет отрицательной, а на шаге 4 условием окончания будет выполнение ограничения* $y^* > y_s$*. Описанные в алгоритме шаги могут выполняться многократно с запоминанием решения, наилучшим образом соответствующего значениям коэффициентам относительной важности и результирующему показателю.*

На рисунке 1.5 приведена иллюстрация работы алгоритма, где каждое перемещение в новую точку сопровождается следующими основными действиями: случайный выбор аргумента из x_1 и x_2; изменение выбранного аргумента таким образом, чтобы значение функции уменьшилось на Δy.

Алгоритм является метаэвристическим, и обеспечивает итерационное улучшение решения благодаря следующим правилам: на каждом шаге значение *y* приближается к целевому с некоторым шагом; выполняется поиск значения аргумента, которое обеспечивает целевое значение функции в итерации.

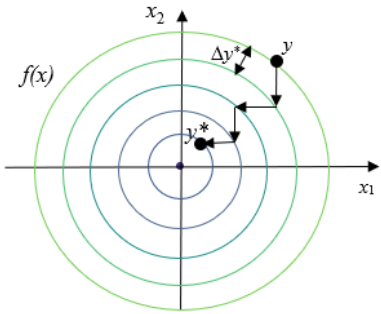


Рис. 1.5. Иллюстрация работы алгоритма

1.3. Решение обратной задачи с использованием регуляризации

Наиболее известными видами регуляризации, основанными на отклонении полученного решения от исходного, являются регуляризация Тихонова [8–10] и регуляризация через манхэттенское расстояние [11, 12], которые также называются L2 и L1 регуляризацией соответственно.

Обратная задача, показанная на рисунке 1.2, может быть представлена в виде операторного уравнения [13]

$$f(x) = y^*. \tag{1.1}$$

Решение уравнения (1.1) может быть определено с помощью минимизации невязки [14]

$$\left\| f(x) - y^* \right\| \to \min. \tag{1.2}$$

Поскольку в рассматриваемой задаче (рисунок 1.2) y^* – число, а x – n-мерный вектор входных значений, то задача (1.2) имеет множество решений.

В случае использования регуляризации по Тихонову обратная задача (рисунок 1.2) может быть представлена в следующем виде (μ – параметр регуляризации) [15]

$$Q(x) = (f(x) - y^*)^2 + \mu \sum_{i=1}^{n} x_i^2 \to \min. \tag{1.3}$$

где Q – минимизируемый функционал.

Здесь минимизируемая функция включает две составляющие: первая компонента отражает соответствие результирующего показателя целевому значению, а вторая – величину аргументов. Решение задачи (1.3) требует нахождение параметра регуляризации μ [15–17]. При решении обратной задачи (1.3) аргументы x определяются таким образом, чтобы их значения были как можно ближе к нулю.

При применении регуляризации через манхэттенское расстояние вместо суммы квадратов аргументов в формуле (1.3) используется сумма модулей аргументов [14]

$$Q(x) = (f(x) - y^*)^2 + \mu \sum_{i=1}^{n} \left| x_i \right| \to \min. \qquad (1.4)$$

В результате решения задачи (1.4) происходит обнуление некоторых аргументов, что может способствовать отбору признаков.

Задача (1.1) может быть представлена в виде задачи условной оптимизации [14]. При этом целевая функция отражает вид регуляризации: минимизация суммы модулей изменений аргументов, минимизация суммы квадратов изменений аргументов.

Так, при минимизации суммы квадратов аргументов задача имеет вид:

$$\begin{aligned} &g(x) = \sum_{i=1}^{n} x_i^2 \to \min, \\ &f(x) = y^*. \end{aligned} \qquad (1.5)$$

где $g(x)$ – целевая функция, отражающая регуляризацию в виде суммы квадратов аргументов.

В случае минимизации суммы модулей аргументов задача имеет вид

$$\begin{aligned} &g(x) = \sum_{i=1}^{n} \left| x_i \right| \to \min, \\ &f(x) = y^*. \end{aligned} \qquad (1.6)$$

Таким образом, обратная задача при использовании регуляризации может быть представлена в виде задачи условной либо безусловной оптимизации.

1.3.1. Метод решения обратной задачи с L1-регуляризацией

Рассмотрим задачу (1.6) применительно к определению изменений аргументов [18–19]:

$$g(\Delta x) = \sum_{i=1}^{n} \left|\Delta x_i\right| \to \min, \qquad (1.7)$$

$$f(x + \Delta x) = y + \Delta y = y^*.$$

В зависимости от вида модели и прироста результата у аргументов целевой функции (1.7) при раскрытии модуля будет либо положительный, либо отрицательный знак.

После раскрытия модуля в случае линейного ограничения ($f(x+\Delta x) = c_1(x_1+\Delta x_1) + c_2(x_2+\Delta x_2) + ... + c_n(x_n+\Delta x_n)$) (c – числовые значения при аргументах в ограничении) задача (1.7) представляет собой задачу линейного программирования. Классическим методом её решения является симплекс-метод. Однако при единственном ограничении и равенстве числовых значений при аргументах в целевой функции единице задача может быть решена более простым методом. Её решение сводится к нахождению элемента с большим абсолютным числовым значением c при приросте аргумента в ограничении и решения уравнения относительно этого аргумента [20].

В случае если ограничение имеет нелинейный вид, рассматриваемая задача относится к задачам нелинейного программирования. Для преобразования ограничения в линейный может быть использовано разложение в ряд Тейлора ($f(\Delta x_1, \Delta x_2, ..., \Delta x_n) = \sum_{i=1}^{n} \frac{\partial f}{\partial \Delta x_i} \Delta x_i$).

Следовательно, в качестве числовых значений при приращениях аргументов рассматривается значение частных производных ($c_i = \dfrac{\partial f}{\partial \Delta x_i}$) в нулевой точке.

Алгоритм решения задачи при минимизации суммы абсолютных значений изменений аргументов включает следующие шаги:

Шаг 1. Вычислить величины c_i и определить номер аргумента k, для которого величина c_k максимальна ($c_k = \max\limits_i \{c_i\}$).

Шаг 2. Выполнить решение уравнения относительно аргумента с номером k

$$f(x_k + \Delta x_k) = y + \Delta y \ .$$

Однако в нелинейных моделях может происходить изменение соотношения производных, поэтому задачу необходимо решать итерационно путем разбиения задачи на подзадачи, в каждой из которых происходит изменение аргумента, имеющего наибольшее значение производной. Итерационный алгоритм решения такой задачи с параметром шага α может быть представлен в следующем виде:

Шаг 1. Используя исходные данные, вычислить функцию ограничения $f(x)$ и сравнить с целевым значением $y + \Delta y$:

Если $f(x) < y + \Delta y$, то изменение аргументов необходимо выполнить в направлении увеличения величины функции ограничения (направлении вектора-градиента): t=1.

Если $f(x) > y + \Delta y$, то изменение аргументов необходимо выполнить в направлении уменьшения величины функции ограничения (направлении вектора-антиградиента): $t = -1$.

Шаг 2. Определить абсолютную разницу между величиной функции ограничения и целевым значением $y + \Delta y$

$$d_0 = \left| f(x) - y - \Delta y \right|.$$

Шаг 3. Вычислить величины c_i и определить номер аргумента k, для которого величина c_k максимальна ($c_k = \max\limits_i \{c_i\}$).

Определить новое значение аргумента путем движения в сторону градиента/антиградиента

$$\Delta x_k^* = \Delta x_k + t \cdot \alpha \cdot c_k .$$

Шаг 4. Вычислить значение функции ограничения $f(x_i^*)$ и отклонение d_1 от заданного значения $y + \Delta y$.

Проверка: если $d_1 > d_0$ или d_1 меньше заданной точности, то работа алгоритма завершается. Иначе $d_0 = d_1$, $\Delta x_k = \Delta x^*_k$, переход на шаг 3.

Решением задачи будут значения Δx.

2. АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ СТОХАСТИЧЕСКОГО МЕТОДА РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ

2.1. Глобальная оптимизация функции

Глобальная оптимизация является важной областью прикладной математики и компьютерных наук и направлена на поиск наилучшего решения из всех возможных вариантов [21]. В отличие от задач локальной оптимизации, целью которых является определение минимума или максимума в определенной окрестности, глобальная оптимизация направлена на определение минимума или максимума во всей области функции. Задачи глобальной оптимизации широко распространены в самых разных областях, включая машинное обучение, где функции часто характеризуются мультимодальностью, нелинейностью и сложными рельефными структурами [22].

Существует два основных подхода к решению задач оптимизации, каждый из которых имеет свои преимущества и ограничения. В первом подходе используются градиентные методы, которые используют значения частных производных

целевой функции [23]. Однако, в случае мультимодальных функций, этот метод решения может привести к определению локального минимума вместо глобального. Тем не менее, этот метод широко используется, особенно в приложениях машинного обучения, поскольку он эффективно находит решения в пространствах большой размерности.

Второй подход заключается в использовании метаэвристических методов, которые выполняют итеративное исследование пространства поиска с использованием эвристик для выявления субоптимальных решений [24]. Коллекция метаэвристических методов обширна и постоянно пополняется новыми вариациями из-за отсутствия универсального оптимального метода (что подтверждается теоремой об отсутствии бесплатных завтраков [25]). Среди основных категорий методов можно выделить эволюционные [26], роевые [27–28], подходы, заимствованные из математики, химии и физики [29], а также методы, основанные на поведении человека [30–31]. Несмотря на эффективность метаэвристических методов их производительность заметно снижается с увеличением размерности задачи, что создает значительные трудности для их применения.

2.1.1. Стохастический алгоритм глобальной оптимизации

Рассмотрим минимизацию функции f с числом аргументов, равном n, выраженную следующим образом:

$$\min_{x \in R^n} f(x) \tag{2.1}$$

Допущения включают непрерывность целевой функции и существование глобального минимума для задачи (2.1). В этом исследовании также предполагается, что функция дифференцируема.

Предлагаемый алгоритм случайного выбора (RC) [32–33] основан на стохастическом методе, представленном в п. 1.2.1: на каждой итерации осуществляется уменьшение значения функции с адаптивным шагом и выполняется решение обратной задачи путем корректировки методом Ньютона выбранного аргумента. При этом

выбор определяется вероятностью β, которая в равной степени применима ко всем аргументам (рисунок 2.1). Однако, если существует априорная информация относительно наиболее эффективных направлений достижения оптимума, эта вероятность может быть настроена индивидуально для каждого аргумента. Каждый аргумент функции связан с показателем u, который характеризует его возможность для включения в расчеты. Этот индикатор принимает одно из двух значений: 1, указывающее на возможность использования аргумента, и 0 при невозможности использования аргумента.

Входные данные алгоритма включают целевую функция $f(x)$, число аргументов n, размер шага для модификации функции Δy, коэффициент уменьшения размера шага r, множитель q для изменения r, верхний предел коэффициента уменьшения шага r_{max}, показатель возможности использования в расчетах u=1, максимальное количество итераций N.

Результатом работы алгоритма являются полученные значения аргументов функции x.

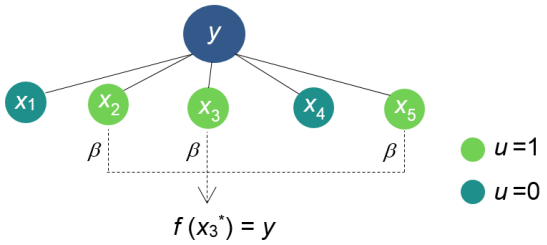


Рис. 2.1. Выбор аргумента для изменения

Алгоритм включает следующую последовательность шагов.

Шаг 1. Случайная генерация начальной точки x, $y = f(x)$.

Шаг 2. Уменьшение функции: $y_{last} = y$, $y = y - \dfrac{\Delta y}{r}$.

Вычисление вероятности выбора:

$$h = \sum_{j=1}^{n} u_j \ .$$

Если $h = 0$, то $r = r \cdot q$. Если $r > r_{max}$, то выход, иначе $u_i = 1$ (для всех $i = 1, n$), $y = y_{last}$ и переход к шагу 2.

Иначе: если $u_i = 1$, то $\beta_i = \dfrac{1}{h}$ иначе $\beta_i = 0$ (i=1, n).

Случайный выбор аргумента в соответствии с вероятностью β (k – номер выбранного аргумента).

Шаг 4. Обновление аргумента.

Сохранить аргументы $x^* = x$. Расчёт x^*_k путем решения уравнения $f(x^*_k) = y$.

Если $f(x^*) < y_{last}$, то $x_k = x^*_k$, $y = f(x^*)$, $u_i = 1$ ($i = 1, n$) иначе $y = y_{last}$, $u_k = 0$.

Переход на шаг 2.

Второй разработанный алгоритм Derivative Choice (DC) основан на вычислении частных производных по каждому аргументу и использовании нормированного значения производной в качестве вероятности. Таким образом, расчёт вероятности β будет выполняться по формулам:

$$c_m = \frac{\partial f}{\partial x_m}, m = 1..n \ ,$$

$$\beta_m = \frac{c_m \cdot u_m}{\sum_{j=1}^{n} c_m \cdot u_m} \ .$$

В данном алгоритме сохраняется элемент случайности при выборе аргумента, однако включается влияние производной, что может быть полезно при локальном поиске минимума.

2.1.2. Результаты экспериментов

Рассмотрим решение глобальной задачи оптимизации функции Растригина в диапазоне от –100 до 100. Отличительной особенностью этой функции является наличие многочисленных локальных минимумов и единственного глобального минимума,

расположенного в точке $x = 0$, где функция достигает значения 0 (рисунок 2.2). Для сравнения эффективности разработанного алгоритма были выбраны следующие методы, реализованные в библиотеках Python:

• Basin-hopping: двухэтапный метод, который объединяет случайный поиск с локальной минимизацией на каждом шаге.

• Алгоритм оптимизации роя частиц (PSO): метод оптимизации, решающий задачу путем итеративной корректировки популяции частиц в пространстве поиска на основе их текущего положения, скорости и значения целевой функции.

• Генетический алгоритм (GA): метаэвристика, основанная на механизмах эволюции, таких как отбор, скрещивание и мутация.

• Двойной отжиг (DA): глобальный алгоритм оптимизации, основанный на имитации отжига и локальном поиске.

Для реализации алгоритмов PSO и GA использовались библиотеки PySwarms и Deap соответственно, методы Dual Annealing и Basin-hopping были реализованы с помощью модуля scipy.optimize. Для алгоритмов Basin-hopping, PSO, DA были использованы гиперпараметры по умолчанию. Для генетического алгоритма установлены следующие параметры: population size = 500, number of generations = 700, crossover probability = 0,9, mutation probability = 0.2. Для методов RC и DC изменялось только максимальное значение параметра уменьшения шага $r_{\max}$: для функции Розенброка оно было установлено в значении 10^8, а для остальных функций – 10^{18}. Остальные гиперпараметры были неизменны: шаг Δy равен 100, коэффициент уменьшения шага r=1, параметр изменения шага уменьшения q=2.

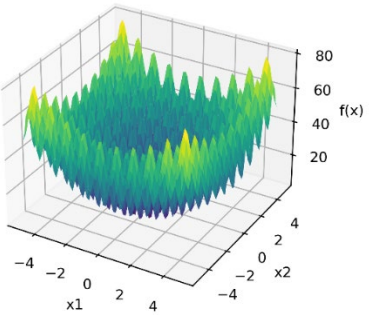


Рис. 2.2. Функция Растригина

В таблице 2.1 представлены результаты экспериментов за 100 случайных реализаций при изменении числа аргументов функции Растригина с 5 до 200. В качестве критериев оценки алгоритмов приведено среднее значение функции и время решения задачи. Согласно полученным результатам, разработанные алгоритмы RC и DC обеспечили лучшие значения критериев для 5 и 10 аргументов. С увеличением количества аргументов алгоритмы показали более низкие значения функции по сравнению с остальными, но затратили больше времени. Наилучшим из рассмотренных метаэвристических алгоритмов оказался DA, т.к. он обеспечил меньшее значение оптимизируемой функции. На рисунке 2.3 показан процент успешного определения минимума (success rate) для случая с 10 аргументами. При этом успешным нахождением минимума функции считалось такое решение, в котором абсолютное отклонение значения функции от глобального минимума не превышало 0,1.

В таблице 2.2 представлены результаты решения задач оптимизации для различных тестовых функций CEC 2022 Benchmark [34]. Критерии оценки включают минимальное значение целевой функции (Min), среднее значение функции (Mean) и среднеквадратичное отклонение функции (STD). В первом столбце указаны названия функций и интервалы поиска. Число аргументов функции было установлено равным 10. В некоторых экспериментах генетический алгоритм не смог решить поставленные задачи; таким образом, результаты были пропущены. Все методы успешно оптимизировали простую функцию Захарова. Для оптимизации функции Розенброка в условиях сложного ландшафта наилучшие результаты были достигнуты с помощью Basin-hopping и DA. Для функции Drop-Wave разработанные RC- и DC-алгоритмы продемонстрировали наихудшую производительность.

Таблица 2.1

Результаты для функции Растригина

Число аргументов	Метрики	Basin-hopping	PSO	GA	DA	RC	DC
5	Mean	0,45	5,85	0,064	$3{,}2\cdot10^{-14}$	$3{,}14\cdot10^{-14}$	$1{,}77\cdot10^{-13}$
	t, c.	0,37	0,25	11,72	0,59	0,19	0,19
10	Mean	2,09	29,85	0,157	0,02	$6{,}39\cdot10^{-12}$	$4{,}55\cdot10^{-13}$
	t, c.	2,92	9,91	16,24	1,81	1,04	0,88
50	Mean	461,04	12,987,67	423,41	0,119	$7{,}69\cdot10^{-10}$	$4{,}5\cdot10^{-11}$
	t, c.	86	1,2	52,2	7,98	52,5	45,5
100	Mean	255,70	56,407,38	1,047,9	0,3	$1{,}24\cdot10^{-9}$	$1{,}47\cdot10^{-9}$
	t, c.	933	2,07	96,4	18,9	403,8	284,4
200	Mean	18,286,25	170,645,83	158,577,07	0,2	$9{,}96\cdot10^{-9}$	$5{,}39\cdot10^{-9}$
	t, c.	3,431	4	191	51,3	1997	1606

Таблица 2.2

Результаты для тестовых функций

Число аргументов	Метрики	Basin-hopping	PSO	DA	RC	DC
Zakharov [–100, 100]	Min	$1{,}68\cdot10^{-16}$	$1{,}13\cdot10^{-7}$	$1{,}42\cdot10^{-14}$	$1{,}18\cdot10^{-14}$	$5{,}68\cdot10^{-14}$
	Mean	$4{,}47\cdot10^{-15}$	1,97	0,02	$1{,}37\cdot10^{-14}$	$1{,}54\cdot10^{-12}$
	STD	$2{,}43\cdot10^{-15}$	5,44	0,14	$1{,}2\cdot10^{-15}$	$2{,}3\cdot10^{-12}$
Rosenbrok [–3, 3]	Min	$3{,}58\cdot10^{-11}$	4,33	$2{,}35\cdot10^{-11}$	$3{,}6\cdot10^{-3}$	$1{,}6\cdot10^{-3}$
	Mean	$5{,}64\cdot10^{-11}$	4058,84	0,08	0,37	0,29
	STD	$3{,}64\cdot10^{-12}$	14056,7	0,56	0,78	0,55
Ackley [-33, 33]	Min	1,15	$5{,}75\cdot10^{-6}$	$1{,}02\cdot10^{-8}$	$1{,}82\cdot10^{-14}$	$1{,}46\cdot10^{-14}$
	Mean	18,86	1,02	$1{,}82\cdot10^{-8}$	1,42	1,69
	STD	2,63	0,95	$2{,}92\cdot10^{-9}$	3,56	3,76
Griewank [–600, 600]	Min	$1{,}88\cdot10^{-11}$	20	$1{,}19\cdot10^{-11}$	0	$5{,}69\cdot10^{-4}$
	Mean	0,21	20,17	0,03	0,06	0,07
	STD	0,99	0,12	0,03	0,04	0,04
Schwefel [–500, 500]	Min	473,80	750,21	$1{,}27\cdot10^{-4}$	$1{,}27\cdot10^{-4}$	$1{,}27\cdot10^{-4}$
	Mean	1593,66	1461,23	$1{,}27\cdot10^{-4}$	353,64	364,61
	STD	516,47	319,91	$5{,}26\cdot10^{-9}$	248,04	234,5
Alpine 1 [–10, 10]	Min	$1{,}52\cdot10^{-7}$	$2{,}2\cdot10^{-4}$	$4{,}88\cdot10^{-9}$	$6{,}61\cdot10^{-7}$	$5{,}87\cdot10^{-7}$
	Mean	0,95	0,21	$4{,}87\cdot10^{-5}$	$5{,}53\cdot10^{-6}$	$5{,}81\cdot10^{-6}$
	STD	3,03	0,33	$6{,}84\cdot10^{-5}$	$3{,}21\cdot10^{-6}$	$3{,}19\cdot10^{-6}$
Drop-Wave [−5.12, 5.12]	Min	$1{,}71\cdot10^{-14}$	0,06	$8{,}88\cdot10^{-15}$	0,71	0,52
	Mean	0,64	0,26	0,28	0,89	0,9
	STD	0,43	0,11	0,15	0,06	0,07

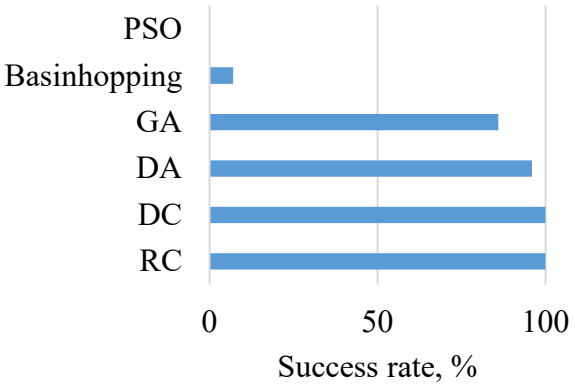


Рис. 2.3. Процент успешного определения минимума при оптимизации функции Расстригина

Таким образом, результаты вычислительных экспериментов демонстрируют возможность использования предложенных алгоритмов для решения задач глобальной оптимизации. Эти алгоритмы обладают рядом преимуществ, в том числе простотой реализации, обусловленной их независимостью от таких сложных концепций, как мутация и скрещивание в генетических алгоритмах или температура в алгоритме имитационного отжига. Кроме того, предлагаемые алгоритмы экономичны с точки зрения памяти, поскольку они не требуют хранения наборов объектов, таких как частицы или популяции. Более того, эти алгоритмы не требуют предварительного определения интервала поиска, который часто может быть неопределенным; вместо этого достаточно определить только начальную точку. Одним из ограничений предлагаемых алгоритмов является то, что они основаны на определении корня функции. Кроме того, изменение только одного аргумента за итерацию может быть неэффективным для функций, включающих плато или сложные ландшафты. Таким образом, предложенные алгоритмы могут быть применены к отдельным задачам, где они обеспечивают оптимальную производительность. В частности, они достигли наилучших результатов при тестировании функции Растригина.

2.2. Обучение полносвязных нейронных сетей

Одним из основных направлений машинного обучения являются искусственные нейронные сети, которые имитируют функционирование человеческого мозга и обладают способностью извлекать полезную информацию из данных, что делает их мощным инструментом для анализа. Приложения, основанные на нейронных сетях, распознают человеческую речь, обнаруживают мошенничество, управляют автономными транспортными средствами и дорожным движением [35], а также прогнозируют экономические показатели [36].

Обучение нейронной сети – это задача оптимизации, характеризующаяся большой размерностью, а также наличием множества локальных минимумов. Классическим подходом к решению этой задачи является градиентный спуск и его различные модификации: стохастический градиентный спуск и адаптивный градиентный спуск (например, Adagrad, Adadelta, RMSProp, Adam и т.д.) [37–39], а также их улучшенные варианты. Однако эти методы имеют следующие ограничения: высокая зависимость от первоначально сгенерированных значений веса и риск застревания в локальном минимуме. Для устранения этих недостатков в литературе предлагается применение метаэвристических алгоритмов, таких как генетические алгоритмы [40], имитационный отжиг и другие. Однако из-за большого количества параметров, подлежащих оптимизации, решение задачи с использованием эвристических алгоритмов требует значительных вычислительных ресурсов. Таким образом, разработка новых методов обучения нейронных сетей и их последующее исследование является актуальной задачей. В данном разделе рассматривается применение разработанных RC- и DC-алгоритмов для обучения нейронных сетей.

Структура нейронной сети включает в себя входные данные x, нейроны скрытого слоя (l – выход нейрона второго слоя), выходные нейроны ($\hat{y}$ – предсказание модели), функции активации f и искомые весовые коэффициенты w (рисунок 2.4). Смещение также учитывается при расчетах путем добавления столбца единиц как к массиву признаков x, так и к массиву скрытых слоев l. В этом исследовании рассматриваются эксперименты, проведенные с использованием однослойной нейронной сети и нейронной сети с

одним скрытым слоем, которые часто используются при анализе данных [41].

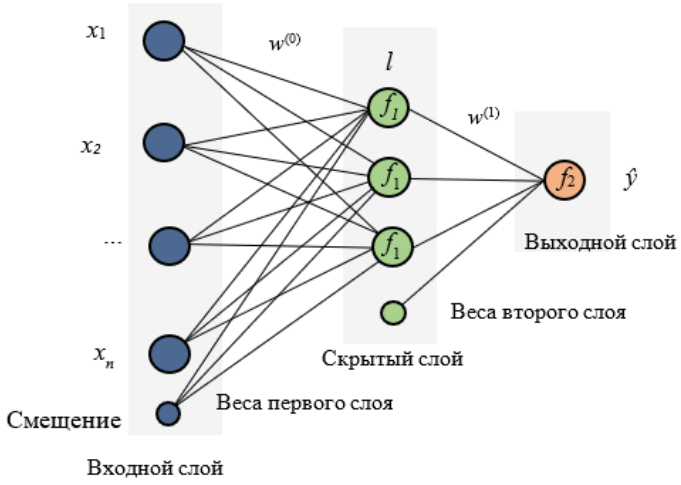


Рис. 2.4. Структура нейронной сети

2.2.1. Алгоритмы обучения нейронной сети

Для обучения сети необходимо оптимизировать функцию J, которая отражает отклонение прогнозируемых значений от фактических:

$$J(w) = \sum_{i=1}^{n} \left(\hat{y}_i - y_i \right)^2 ,$$

где y – вектор фактических выходных значений;
$\hat{y}$ – вектор прогнозируемых значений;
n – количество наблюдений.

Входные данные алгоритма:

- X – матрица входных значений (включая смещение);
- y – выходные значения;
- input_neurons – количество входных нейронов;
- hidden_neurons – количество нейронов скрытого слоя;
- d – делитель, определяющий размер шага ΔJ;
- r – коэффициент уменьшения размера шага;

• r_{max} – верхний предел коэффициента уменьшения шага;

• u – вектор активности размер($w^{(0)}$) + размер($w^{(1)}$), инициализируемый единицами.

Выходные данные:

• $w^{(0)}$ – веса первого слоя (вход → скрытый слой);

• $w^{(1)}$ – веса второго слоя (скрытый слой → выходной слой).

Начальные значения $w^{(0)}$ и $w^{(1)}$ инициализируются случайными величинами с нормальным распределением: $w^{(0)}$ = случайная_матрица(input_neurons+1, hidden_neurons), $w^{(1)}$ = случайный_вектор(hidden_neurons+1).

Для оптимизации параметров нейронной сети алгоритм RC будет включать следующие шаги [33, 42].

Шаг 1. Инициализация. Веса w инициализируются случайным образом, и вычисляется функция ошибки, а также шаг ее уменьшения:

$$l = f_1\left(X \cdot w^{(0)}\right),$$

$$\hat{y} = f_2\left(l_{bias} \cdot w^{(1)}\right),$$

$$J(w) = \sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2, \tag{2.2}$$

$$J_{prev} = J(w),$$

$$\Delta J = \frac{J(w)}{d},$$

где l_{bias} – выход скрытого слоя со смещением.

Шаг 2. Уменьшение функции ошибки на величину шага:

$$J = J_{prev} - \frac{\Delta J}{r}.$$

Шаг 3. Выбор весового коэффициента для модификации:

Для каждого веса с u = 1 вычислить вероятность выбора β (m=1, число весов):

$$h = \sum_{j=1}^{n} u_j ,$$

$$\beta_m = \begin{cases} \dfrac{1}{h} & если \quad u_m = 1, \\ 0 & иначе. \end{cases}$$

Выполнить выбор весового коэффициента с индексом p, исходя из вероятности β. Если выбор пуст, увеличить коэффициент уменьшения шага $r = r \cdot q$ (q – коэффициент изменения шага). Если r превышает максимальное пороговое значение $r_{\max}$, алгоритм завершается. В противном случае установить значение u для индикатора использования равным единице для всех весовых коэффициентов w, пересчитать вероятность β и выполнить случайный выбор весового коэффициента на основе обновленных вероятностей выбора β.

Шаг 4. Если выбранный весовой коэффициент относится ко второму уровню ($p \geq$= ((число input_neurons + 1) × hidden_neurons)), перейти на шаг 5; в противном случае – на шаг 6.

Шаг 5. Корректировка весового коэффициента, относящегося ко второму уровню.

Вычислить индекс выбранного коэффициента v в пределах уровня:

$$v = p - (\text{input_neurons} + 1) \times \text{hidden_neurons}.$$

Вычислить градиент g_1:

$$y_{error} = \hat{y} - y ,$$

$$y_{delta} = y_{error} \odot f_2'(\hat{y}),$$

$$g_1 = l_{bias}{}^T y_{delta}, \tag{2.3}$$

где $f_2'(\hat{y})$ - производная функции активации;

$\odot$ – поэлементное умножение.

Сформировать новый массив весовых коэффициентов, чтобы обеспечить возможность последующей пробной модификации выбранного элемента:

$$w^{*(1)} = w^{(1)} .$$

Обновить выбранный вес $w^{(1)}{}_v$ путем решения уравнения:

$$J(w^{*(1)}{}_v) = J\,.$$

Итерационная формула Ньютона имеет вид:

$$w^{*(1)}{}_v = w^{*(1)}{}_v - \frac{J(w^*) - J}{2g_{1v}}\,.$$

Значения для $J\,(w^*)$ и g_{1v} пересчитываются после каждой итерации с использованием уравнений (2.2) и (2.3).

Новое значение функции ошибки вычисляется с использованием полученного весового коэффициента:

$$J_{new} = J(w^*)\,.$$

Если $J_{new} < J_{prev}$, тогда $w^{(1)}=w^{*(1)}$, u=1 для всех весов, и $J_{prev}= J_{new.}$ Переход к шагу 2.

Иначе u_p = 0. Переход к шагу 2.

Шаг 6. Корректировка весового коэффициента, относящегося к первому уровню:

Определить индексы строк и столбцов v и b для матрицы коэффициентов первого уровня путем разложения линейного индекса p:

$$v = p \bmod \text{hidden_neurons}$$
$$b = \lfloor p/\text{hidden_neurons} \rfloor$$

Ошибка скрытого слоя и градиент вычисляются следующим образом:

$$l_{error} = y_{delta} \cdot w^{(1)}{}_{[:-1]}{}^T\,,$$

$$l_{delta} = l_{error} \odot f_1'(l)\,.$$

$$g_0 = X^T l_{delta}\,. \qquad (2.4)$$

где $w^{(1)}$[:-1] – это массив весовых коэффициентов $w^{(1)}$ за исключением смещения.

Массив w_0 копируется в массив $w^{*(0)}$: $w^{*(0)} = w^{(0)}$.

Выбранный вес $w_{0b,v}$ обновляется путем решения уравнения:

$$J(w^{*(0)}{}_{b,v}) = J\,.$$

Для решения уравнения методом Ньютона используется итерационная формула следующего вида:

$$w^{*(0)}{}_{b,v} = w^{*(0)}{}_{b,v} - \frac{J(w^*) - J}{4g_{0b,v}}.$$

$J(w^*)$ и $g_{0b,v}$ пересчитываются по формулам (2.2) и (2.4).

Вычисляется новое значение функции ошибки с учетом обновленных весов:

$$J_{new} = J(w^*).$$

Если $J_{new}<J_{prev}$, тогда $w^{(0)}=w^{*(0)}$, u=1 для всех весов, и $J_{prev}= J_{new.}$ Переход к шагу 2.

Иначе u_p = 0. Переход к шагу 2.

В случае применения алгоритма RC вероятности на шаге 3 равны, при использовании DC – пропорциональны абсолютным значениям частных производных. Для определения этих вероятностей из вектора u первоначально формируются массивы c_0 и c_1, соответствующие массивам w_0 и w_1 и содержащие индикаторы возможности использования каждого элемента в расчетах. Далее определяются абсолютные значения градиентов с учетом индикаторов возможности использования:

$$cw_0 = |g_0| \odot c_0,$$

$$cw_1 = |g_1| \odot c_1.$$

Наконец, вероятности выбора весового коэффициента определяются по формуле:

$$\beta_0 = \frac{cw_0}{\sum cw_0 + \sum cw_1},$$

$$\beta_1 = \frac{cw_1}{\sum cw_0 + \sum cw_1},$$

где β_0 – матрица, включающая вероятности выбора весовых коэффициентов первого слоя;

β_1 – вектор, содержащий вероятности выбора весовых коэффициентов второго слоя.

Для однослойной сети выполняется расчет только матрицы β_0.

2.2.2. Результаты экспериментов

Для проведения вычислительных экспериментов с однослойной нейронной сетью были использованы данные из хранилища KEEL, включая такие наборы данных, как treasury, stock, autoMPG8, baseball, concrete и machineCPU. Входные и выходные переменные были стандартизированы в ходе предварительной обработки. Рассматривались как классические функции активации (линейные), так и более сложные, в том числе предложенные для моделирования временных рядов [43]: snake, sincos, ISRU, actg, ln, softsign, th, softplus и cloglogm. Кроме того, функция Химмельблау, которая широко используется в качестве эталона в задачах глобальной оптимизации, также рассматривалась как функция активации.

Кроме разработанных алгоритмов оптимизации, RC и DC, были рассмотрены классические методы, широко используемые для обучения нейронных сетей, такие как Adam и SGD, а также метаэвристический подход DA. Эксперименты проводились в течение 10 случайных прогонов. Для методов Adam и SGD скорость обучения варьировалась от 0,00001 до 1, и выбирался наилучший вариант. Для алгоритмов RC и DC гиперпараметры остались неизменными: в частности, шаг ΔJ был равен исходному значению функции J, деленному на 10, максимальный параметр уменьшения шага $r_{\max}$ составлял 100 000, коэффициенты для изменения шага r и q равны 1 и 2 соответственно. Начальные значения весовых коэффициентов были сгенерированы в диапазоне [0; 0,01].

На основании результатов, представленных в таблице 2.3, можно сделать следующие выводы. DA продемонстрировал один из лучших результатов в решении задач глобальной оптимизации функции (п.2.1.2), однако при обучении нейронных сетей в большинстве сценариев он уступает другим методам. Наилучшая производительность, достигнутая с помощью этого метода, была получена при использовании функции активации sincos. Также методы DC и RC для этой функции активации продемонстрировали более высокую эффективность, чем Adam и SGD. Метод DC показал наилучшие результаты по среднеквадратичной ошибке (MSE) и средней абсолютной ошибке (MAE) в большинстве сценариев; однако методы Adam и SGD превзошли RC и DC по времени вычис-

лений (рисунок 2.5). Сравнение MSE и MAE (таблица 2.4) показывает, что разработанные алгоритмы обеспечили наилучшие результаты на следующих наборах данных: concrete, AutoMPG8 и machineCPU.

Таблица 2.3

Число наилучших вариантов при разных активационных функциях

Набор данных	Число активационных функций, при которых метод показал наилучшие результаты по mse					Число активационных функций, при которых метод показал наилучшие результаты по mae				
	Adam	SGD	RC	DC	DA	Adam	SGD	RC	DC	DA
treasury	1	0	2	7	1	2	0	0	8	1
stock	0	0	7	3	1	1	0	7	2	1
autoMPG8	0	1	3	7	0	0	0	5	6	0
baseball	0	3	4	3	0	0	3	4	3	0
concrete	1	0	2	7	1	1	0	5	5	0
machineCPU	3	5	1	2	0	1	1	4	5	0

Таблица 2.4

Наилучшие результаты по MSE

Набор данных	Adam		SGD		RC		DC		DA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
treasury	$6{,}2\cdot 10^{-4}$	$4{,}7\cdot 10^{-3}$	$9\cdot 10^{-4}$	$6{,}1\cdot 10^{-3}$	$6{,}5\cdot 10^{-4}$	$5\cdot 10^{-3}$	$6{,}5\cdot 10^{-4}$	$4{,}7\cdot 10^{-3}$	0,005	0,048
stock	0,0128	0,0273	0,013	0,028	0,01278	0,0274	0,0128	0,0274	0,036	0,158
autoMPG8	0,0183	0,0319	0,0178	0,031	0,0175	0,0301	0,0174	0,03	0,023	0,108
baseball	0,0378	0,0484	0,0376	0,0482	0,0384	0,0486	0,0386	0,0487	0,765	0,7
concrete	0,0343	0,0463	0,0342	0,0463	0,0336	0,0453	0,0335	0,0452	0,02	0,113
machineCPU	0,0421	0,0365	0,042	0,0365	0,042	0,0364	0,0418	0,0363	0,427	0,587

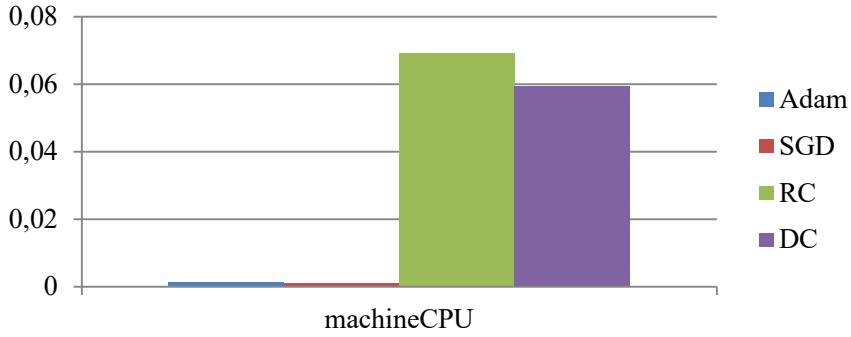


Рис. 2.5. Время решения задачи для набора данных machineCPU

Алгоритмы DA, RC и DC продемонстрировали более эффективное обучение модели при использовании активации sincos, чем классические методы; таким образом, была рассмотрена возможность их применения к задаче моделирования временных рядов, которые часто характеризуются циклическими изменениями. В качестве тестового набора данных был использован температурный временной ряд, полученный с ирландской метеостанции [44]. Набор данных включает 6075 ежедневных наблюдений с 11 августа 2003 года по 31 мая 2020 года, причем последние 1500 наблюдений использовались в качестве тестового набора. Для обучения нейронной сети были использованы следующие данные:

• Выходная переменная: максимальная температура (Maxtp).

• Входные переменные: cbl – среднее скорректированное давление на уровне барометра (hPa), wdsp – средняя скорость ветра (kt), hm – максимальная десятиминутная средняя скорость ветра (kt), maxtp_s – максимальная температура на предыдущем временном шаге.

Исходные значения весов были сгенерированы в диапазоне от 0 до 1.

На рисунке 2.6 показаны фактические значения для тестового набора данных и прогнозируемые значения, полученные с использованием RC-алгоритма. На рисунке 2.7 представлена диаграмма значений MSE, которая была построена на основе тестового набора

данных для 100 случайных реализаций. Метод RC имел наименьший разброс и средние значения. В то же время наибольший разброс наблюдался для методов Adam и DA.

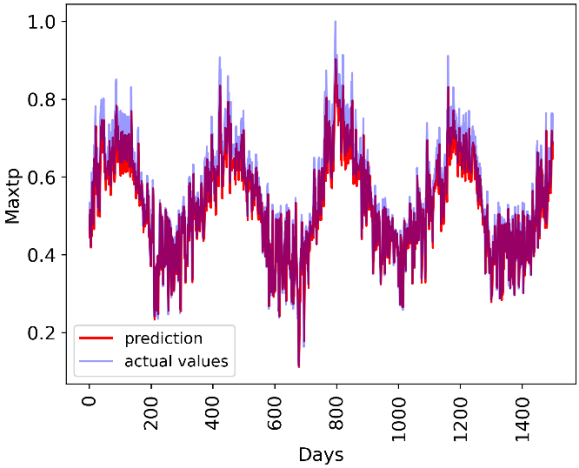


Рис. 2.6. Реальные (actual values) и предсказанные (prediction) значения для тестовой выборки (алгоритм RC)

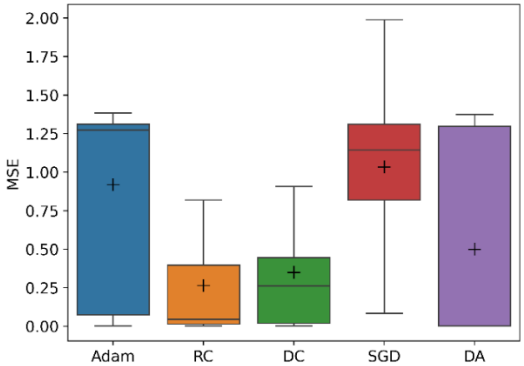


Рис. 2.7. Boxplot для MSE для различных методов на тестовой выборке. Здесь плюс – среднее значение, горизонтальная линия – медиана, прямоугольник — квартильный диапазон от 25 до 75%, усы – минимальное и максимальное значения

Также было рассмотрено обучение нейронной сети с одним скрытым слоем, содержащим два нейрона. Для активации скрытого слоя была использована функция sincos, а выходного слоя – функция Растригина с нормированием, равным 5000. В этом эксперименте исследовалось, как методы работают на сложных и нестандартных активационных функциях.

Данные для экспериментов были основаны на общем рыночном индексе Wilshire 2500 (https://www.wilshireindexes.com), ключевом индикаторе фондового рынка США, который отражает динамику акций со средней капитализацией. Для обучения были использованы ежедневные данные с 4 января 2021 года по 29 декабря 2023 года, включающие 754 наблюдений. В качестве тестового набора были использованы последние 200 наблюдений. Для обучения нейронной сети были использованы следующие данные:

• Выходные переменные: значения индекса Wilshire в день *t*.

• Входные переменные: значения индекса Wilshire в дни *t*-1, *t*-2, *t*-3 и *t*-4.

Boxplot диаграмма для MSE на тестовом наборе данных представлена на рисунке 2.8, на котором показано, что методы RC и DC демонстрируют наименьшую дисперсию, а также самые низкие средние и медианные значения. Метод DA оказался наиболее время затратным, продолжительность его выполнения для решения задачи составила 46,8 секунд. Для алгоритмов RC, DC, Adam и SGD среднее время выполнения составило 4,83, 3,19, 2,77 и 1,53 секунд соответственно.

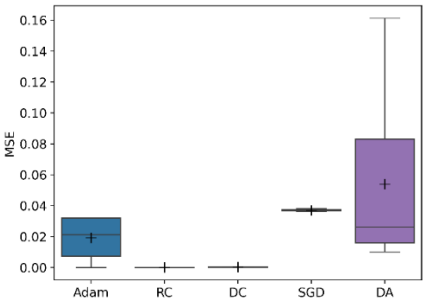


Рис. 2.8. Диаграмма MSE
для различных методов на тестовой выборке

2.3. Кластеризация данных

Кластеризация – фундаментальный метод обучения без учителя, широко применяемый в биоинформатике, анализе изображений и других областях для структурирования данных в условиях априорной неопределенности [45–48]. Целью кластеризации является разбиение совокупности объектов на группы (кластеры) таким образом, чтобы объекты внутри одной группы были более схожи между собой, чем с объектами из других групп. Выбор конкретного алгоритма кластеризации и его параметров критически влияет на интерпретацию результатов и последующие выводы, особенно при работе с многомерными или зашумленными данными. Существующие методы можно разделить на две широкие категории: жесткая кластеризация, приписывающая каждый объект строго к одному кластеру [49, 50], и мягкая (нечеткая) кластеризация, определяющая степень принадлежности объекта всем кластерам [51–54].

Переход к мягкой кластеризации сглаживает целевую функцию, делая ее пригодной для градиентного спуска, в то время как применение градиентных методов к целевой функции жесткой кластеризации затруднено из-за ее кусочной структуры и разрывности. Вместе с тем функция сохраняет множество локальных минимумов и характеризуется высокой размерностью, что ограничивает эффективность стандартных оптимизаторов: градиентных методов и метаэвристик [55]. Таким образом, разработка алгоритма мягкой кластеризации устойчивого к локальным минимумам, остается актуальной проблемой.

2.3.1. Алгоритм кластеризации

Пусть задано множество объектов $X = \{x_1, x_2, ..., x_N\} \subset R^D$, которое необходимо разбить на K кластеров. Каждый k-й кластер характеризуется центром $c_k \in R^D$. В отличие от методов жесткой кластеризации, алгоритмы мягкой (нечеткой) кластеризации вводят для каждого объекта x_i вектор степеней принадлежности

$w_i \in [0,1]^k$, где элемент w_{ik} интерпретируется как мера уверенности в том, что объект x_i относится к кластеру с центром c_k. В данной работе предлагается целевая функция следующего вида:

$$L(C,S) = \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ik} \left\| x_i - c_k \right\|^2, \tag{2.5}$$

где веса w_{ik} вычисляются с использованием механизма softmax, параметризованного вектором чувствительности (масштаба) $S = (s_1, s_2, ..., s_k) \in \mathbb{R}_+^K$:

$$w_{ik} = \frac{\exp\left(-s_k \left\| x_i - c_k \right\|^2\right)}{\sum_{j=1}^{K} \exp\left(-s_j \left\| x_i - c_j \right\|^2\right)}.$$

Здесь $\left\| x_i - c_k \right\|^2$ обозначает квадрат евклидова расстояния между объектом x_i и центром кластера c_k. Параметры s_k играют роль, аналогичную обратной температуре, и определяют "жесткость" границ между кластерами: при $s_k \to \infty$ распределение принадлежности стремится к жесткому (one-hot), а при малых значениях s_k – к равномерному. Оптимизационная задача заключается в нахождении матрицы центров $C \in \mathbb{R}^{K \times D}$ и вектора масштабов $S \in \mathbb{R}_+^K$, минимизирующих функционал $L(C,S)$.

Для решения полученной оптимизационной задачи необходимо модифицировать алгоритм п.2.1.1 следующим образом:

- Численно вычислить производную.
- Спроецировать параметры на область допустимых значений. Для центров кластеров устанавливаются границы, соответствующие минимальному и максимальному значениям признаков в обучающей выборке. Для параметров масштаба устанавливаются эмпирически выявленные границы от 0,01 до 10.
- Повторно инициализировать центры в случае пустых кластеров.

Модель оптимизации (2.5) основана на евклидовом расстоянии, что предполагает сферическую форму кластеров. Следовательно, предложенный алгоритм лучше всего подходит для наборов данных, в которых кластеры имеют выпуклую форму.

2.3.2. Результаты экспериментов

Для оценки эффективности предложенного оптимизационного алгоритма проведено сравнительное исследование с хорошо зарекомендовавшими себя методами кластеризации: k-средних [56] для жёсткой кластеризации, а также FCM [57] и гауссовской смеси (GM) [58] для нечёткой кластеризации. Для оптимизации функции (2.5) использовался метод последовательного квадратичного программирования (SLSQP) [59] и алгоритм дифференциальной эволюции (DE) [60], поскольку оба метода продемонстрировали высокую эффективность на задачах условной мультимодальной оптимизации [33]. Все алгоритмы использовали одинаковую инициализацию центров кластеров с использованием нормального распределения. Реализация алгоритмов выполнена на языке Python с использованием стандартных библиотек для базовых методов: scikit-learn [61] для k-средних и GM, skfuzzy для FCM и SciPy [62] для SLSQP и DE.

Для k-средних, SLSQP и DE использовались параметры по умолчанию. Для FCM параметр нечёткости был задан равным 2, максимальное число итераций – 1000, а допустимая погрешность сходимости – 0.00001. Параметры предложенного алгоритма RC были подобраны эмпирически и сохранялись неизменными во всех экспериментах без дополнительной настройки. В частности, максимальное число итераций внешнего цикла $T_{max} = 2000$, максимальное число итераций внутреннего ньютоновского поиска $Q_{max} = 150$. Начальный шаг снижения уровня целевой функции вычислялся автоматически как $\Delta = L_0/100$, где L_0 – значение целевой функции при начальном наборе параметров. Коэффициент увеличения шага r инициализировался единицей. Для численного дифференцирования использовалось значение $\varepsilon = 10^{-5}$. Внутренний цикл завершался, когда абсолютное значение производной ниже 10^{-10} или изменение параметра становится меньше 10^{-5}. Начальное значение параметра масштаба s_k принималось равным 1,05.

Предложенный алгоритм тестировался в двух различных сценариях. В первом сценарии жёсткая кластеризация выполнялась путём дефаззификации полученного нечёткого разбиения: по завершении оптимизации каждому объекту присваивалась метка кластера, соответствующая максимальной степени принадлежности. Во втором сценарии оценивалось качество нечёткой кластеризации.

Работа предложенного алгоритма в обоих сценариях – жёсткой и нечёткой кластеризации – оценивалась на десяти эталонных наборах данных из репозитория машинного обучения UCI [63], широко используемых для сравнительного анализа алгоритмов кластеризации [64–65]: Iris (отобранные признаки: SepalLengthCm, PetalLengthCm, PetalWidthCm), Banana, Customer Segmentation (отобранные признаки: Age, Income), User Knowledge Modeling, Mall Customers, Glass Identification, Wine, Ecoli и Yeast. Дополнительно был сгенерирован синтетический набор данных (makeblobs) с помощью функции make_blobs библиотеки scikit-learn (300 объектов, 10 кластеров, стандартное отклонение 0,60, random_state=20). Выбранные наборы данных представляют собой данные средней размерности: количество признаков варьируется от 2 до 13, число кластеров – от 2 до 10, а число наблюдений – от 148 до 5300. Ко всем наборам данных на этапе предобработки была применена z-нормализация признаков.

Для каждого метода выполнялось 5 случайных реализаций с различными начальными приближениями, и итоговым результатом выбиралось решение с наименьшим значением целевой функции. Сравнительный анализ проводился по пяти метрикам: значение целевой функции L, индекс Дэвиса–Боулдина (Davies Bouldin, DB), индекс Калински–Харабаша (Calinski Harabasz, CH), коэффициент силуэта (Silhouette), скорректированный индекс Рэнда (Adjusted Rand Index, ARI) и чистота (Purity) (при наличии истинных меток). Результаты экспериментов представлены в таблице 2.5. В таблице 2.6 приведено количество метрик (из числа использованных для каждого набора данных), по которым каждый метод показал наилучший результат.

Таблица 2.5

Значения метрик жесткой кластеризации

Датасет	D	K	Метрика	k-means	FCM	SLSQP	RC	DE
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Iris	3	3	L	–	–	63,21	63,21	63,21
			DB	**0,6848**	0,6884	**0,6848**	**0,6848**	**0,6848**
			CH	451,09	**451,12**	451,09	451,09	451,09
			Silhouette	**0,5404**	0,5385	**0,5404**	**0,5404**	**0,5404**
			ARI	**0,6898**	0,6734	**0,6898**	**0,6898**	**0,6898**
			Purity	**0,8733**	0,8667	**0,8733**	**0,8733**	**0,8733**
Banana	2	2	L	–	–	6102,3	6102,3	6102,3
			DB	**1,0460**	1,0476	**1,0460**	**1,0460**	**1,0460**
			CH	3913,9	3910,9	**3914,2**	**3914,2**	**3914,2**
			Silhouette	0,3888	0,3885	**0,3889**	**0,3889**	**0,3889**
			ARI	0,0165	0,0172	**0,0175**	**0,0175**	**0,0175**
			Purity	0,5647	0,5660	**0,5666**	**0,5666**	**0,5666**
Customer Segmentation	2	2	L	–	–	2262,6	2262,6	2262,6
			DB	**0,9917**	0,9951	**0,9917**	**0,9917**	**0,9917**
			CH	**1538**	1530,3	**1538**	**1538**	**1538**
			Silhouette	**0,4428**	0,4369	0,4426	0,4426	0,4426
			ARI	–	–	–	–	–
			Purity	–	–	–	–	–

Продолжение таблицы 2.5

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
User Knowledge Modeling	5	5	L	–	–	1119,3	1137,3	1119,4
			DB	1,4697	2,3273	1,4583	**1,4539**	1,4584
			CH	79,61	57,97	**79,69**	79,48	79,68
			Silhouette	0,1823	0,1083	**0,1827**	0,1823	0,1814
			ARI	0,0906	**0,2292**	0,1020	0,1123	0,1026
			Purity	0,4665	**0,5161**	0,4566	0,4690	0,4665
Mall Customers	5	5	L	–	–	168,37	168,43	168,37
			DB	**0,8746**	0,8949	**0,8746**	**0,8746**	**0,8746**
			CH	**125,1**	123,51	**125,1**	**125,1**	**125,1**
			Silhouette	**0,4166**	0,4053	**0,4166**	**0,4166**	**0,4166**
			ARI	–	–	–	–	–
			Purity	–	–	–	–	–
Glass Identification	9	6	L	–	–	912,48	816,45	1247,2
			DB	1,0890	1,5118	1,1642	**0,9989**	1,2458
			CH	56,19	42,84	46,34	**61,55**	38,1
			Silhouette	**0,3274**	0,2152	0,3107	0,3075	0,2162
			ARI	0,1492	0,1550	**0,1645**	0,1412	0,0497
			Purity	0,4907	**0,5467**	0,5421	0,5374	0,4299
Wine	13	3	L	–	–	1277,9	1281,9	1658,8
			DB	**1,3892**	**1,3892**	**1,3892**	**1,3892**	1,5260
			CH	**70,94**	**70,94**	**70,94**	**70,94**	69,52
			Silhouette	**0,2849**	**0,2849**	**0,2849**	**0,2849**	0,2593
			ARI	**0,8975**	**0,8975**	**0,8975**	**0,8975**	0,3743
			Purity	**0,9663**	**0,9663**	**0,9663**	**0,9663**	0,6011

Окончание таблицы 2.5

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Makeblobs	2	10	L	–	–	19,23	10,06	10,14
			DB	0,6499	0,5686	0,4626	0,4847	**0,4283**
			CH	1504,1	2740,2	1569,8	**3311,9**	2995,3
			Silhouette	0,6167	0,6491	0,6674	0,6609	**0,6913**
			ARI	0,8491	0,8662	0,7194	**0,9782**	0,8951
			Purity	0,89	0,9	0,7	**0,9900**	0,90
Ecoli	7	8	L	–	–	565,16	584,37	1663,41
			DB	1,0937	1,7580	0,8608	0,8034	**0,4617**
			CH	**158,81**	52,29	148,22	153,54	68,92
			Silhouette	0,2570	0,1784	0,3532	0,3592	**0,5577**
			ARI	0,5335	0,3672	**0,7298**	0,7166	0,0381
			Purity	0,8214	0,8065	**0,8274**	0,8125	0,4435
Yeast	8	10	L	–	–	4517,2	4728	5961,1
			DB	1,0890	2,6331	1,1705	1,2319	**1,0579**
			CH	**316,13**	163,6	300,52	294,24	273,17
			Silhouette	0,2055	0,1015	**0,2223**	0,1909	0,2142
			ARI	0,1823	0,1265	**0,2127**	0,1792	0,0892
			Purity	**0,5344**	0,3389	0,5216	0,5142	0,4036

Таблица 2.6

Количество метрик, по которым каждый метод кластеризации показал наилучший результат

Датасет	k-means	FCM	SLSQP	RC	DE	Общее число метрик
Iris	4	1	4	4	4	5
Banana	1	0	5	5	5	5
Customer Segmentation	3	0	2	2	2	3
User Knowledge Modeling	0	2	2	1	0	5
Mall Customers	3	0	3	3	3	3
Glass Identification	1	1	1	2	0	5
Wine	5	5	5	5	0	5
Makeblobs	0	0	0	3	2	5
Ecoli	1	0	2	0	2	5
Yeast	2	0	2	0	1	5
Итого	20	9	26	25	19	46

Результаты подтверждают эффективность предложенной оптимизационной модели (из 46 пар «набор данных × метрика» SLSQP, RC и DE достигли лучших значений в 57%, 54% и 41% случаев соответственно). При этом SLSQP и RC показали наилучшие значения метрик на большинстве датасетов.

В таблице 2.7 представлены результаты экспериментов для мягкой кластеризации. В качестве метрик качества использовались коэффициент нечеткости разбиения (PC), энтропия классификации (CE), нечеткий силуэт (FSI) и индекс Xie-Beni (XB). Для каждого метода выполнялось 5 случайных реализаций с различными начальными приближениями, и итоговым результатом выбиралось решение с наименьшим значением целевой функции.

По совокупности метрик (таблица 2.8) на большинстве датасетов DE показал лучшие или близкие к лучшим значения PC, CE, FSI и XB. SLSQP продемонстрировал практически идентичные DE результаты на «простых» датасетах, но был менее устойчив на сложных данных. RC показал лучшие результаты, в частности, на Customer Segmentation (PC, CE), User Knowledge Modeling (FSI, XB) и Glass Identification (XB); на Iris он также делит лучший результат по PC и CE с SLSQP и DE.

Таблица 2.7

Значения метрик мягкой кластеризации

Датасет	Метрика	FCM	SLSQP	RC	DE	GM
Iris	PC	0,7724	**0,9901**	**0,9901**	**0,9901**	0,9714
	CE	0,4161	**0,0183**	**0,0183**	**0,0183**	0,0506
	FSI	**0,6897**	0,6554	0,6554	0,6554	0,5885
	XB	**0,1284**	0,1552	0,1552	0,1552	0,2627
Banana	PC	0,7196	**0,9865**	0,9862	**0,9865**	0,8940
	CE	0,4361	**0,0225**	0,0230	**0,0225**	0,1771
	FSI	**0,5549**	0,5207	0,5208	0,5207	0,5069
	XB	**0,2469**	0,3283	0,3281	0,3283	0,3767
Customer Segmentation	PC	0,7490	0,9863	**0,9884**	0,9863	0,8413
	CE	0,3968	0,0228	**0,0194**	0,0228	0,2659
	FSI	**0,5976**	0,5699	0,5692	0,5699	0,5433
	XB	**0,2287**	0,2775	0,2785	0,2775	0,3526

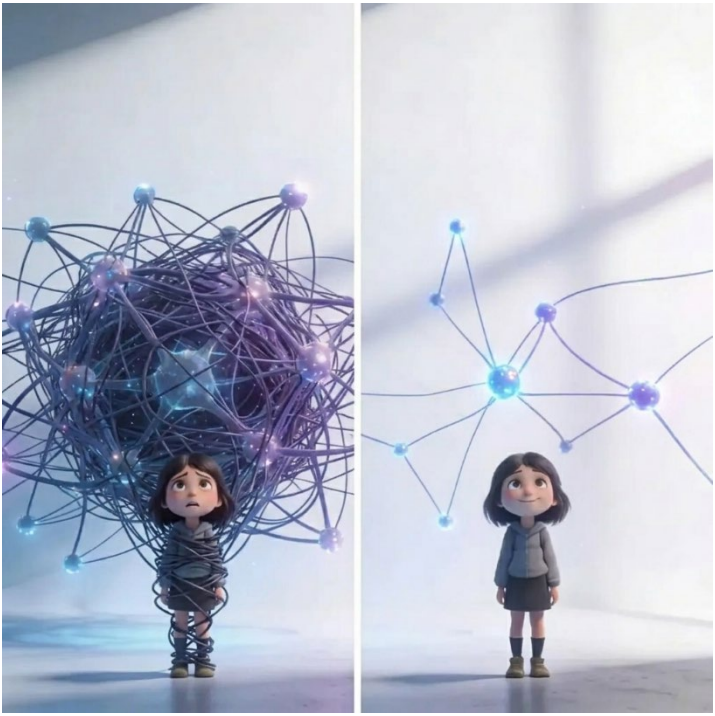
Окончание таблицы 2.7

Датасет	Метрика	FCM	SLSQP	RC	DE	GM
User Knowledge Modeling	PC	0,2037	**0,9911**	0,9076	0,9903	0,8997
	CE	1,6008	**0,0172**	0,1642	0,0181	0,1808
	FSI	0,0722	0,3059	**0,3224**	0,3058	0,2845
	XB	$4,6 \times 10^{11}$	0,6946	**0,5834**	0,6858	0,7754
Mall Customers	PC	0,5173	**0,9958**	0,9941	**0,9958**	0,9600
	CE	0,9958	**0,0085**	0,0115	**0,0085**	0,0750
	FSI	**0,5949**	0,5512	0,5518	0,5513	0,5430
	XB	**0,1914**	0,2821	0,2810	0,2821	0,3159
Glass Identification	PC	0,3703	0,9945	0,9070	**0,9992**	0,9851
	CE	1,3011	0,0105	0,1587	**0,0021**	0,0250
	FSI	0,4046	**0,4789**	0,4783	0,3679	0,3942
	XB	3,6481	0,8574	**0,6355**	1,0535	0,9067
Wine	PC	0,5404	0,9998	0,9779	**0,9999**	0,9947
	CE	0,7891	0,0007	0,0395	**0,0004**	0,0091
	FSI	0,4425	0,4853	0,4921	**0,5178**	0,4165
	XB	1,0212	0,6730	0,6429	**0,4296**	0,5541
Makeblobs	PC	0,7921	0,9748	0,8345	0,9200	**0,9906**
	CE	0,4878	0,0512	0,2737	0,1477	**0,0176**
	FSI	0,7597	0,7503	0,7636	**0,7848**	0,7562
	XB	0,7502	**0,1114**	0,3791	0,1184	0,2061
Ecoli	PC	0,7632	0,9945	0,9268	**1,0000**	0,9676
	CE	0,3966	0,0102	0,1313	**0,0000**	0,0616
	FSI	0,4320	0,5020	0,5301	**0,6722**	0,2478
	XB	0,8314	0,5576	0,4114	**0,1370**	3,5326
Yeast	PC	0,1083	0,9822	0,7958	**1,0000**	0,8750
	CE	2,2646	0,0301	0,3643	**0,0000**	0,2363
	FSI	0,1283	0,3597	0,3502	**0,7594**	-0,0078
	XB	$1,19 \times 10^{13}$	0,7963	0,5980	**0,0670**	4,1152

Таблица 2.8

Количество метрик, по которым каждый метод нечеткой кластеризации показал наилучший результат

Датасет	FCM	SLSQP	RC	DE	GM	Общее число метрик
Iris	2	2	2	2	0	4
Banana	2	2	0	2	0	4
Customer Segmentation	2	0	2	0	0	4
User Knowledge Modeling	0	2	2	0	0	4
Mall Customers	2	2	0	2	0	4
Glass Identification	0	1	1	2	0	4
Wine	0	0	0	4	0	4
Makeblobs	0	1	0	1	2	4
Ecoli	0	0	0	4	0	4
Yeast	0	0	0	4	0	4
Итого	8	10	7	21	2	40



3. АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ НА ОСНОВЕ МЕТОДА РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ С L1-РЕГУЛЯРИЗАЦИЕЙ

3.1. Обучение LASSO-регрессии

Востребованность регрессионного анализа определяется способностью регрессионных моделей обеспечивать интерпретируемость результата и доступностью статистически обоснованных инструментов для оценки значимости функций и коэффициентов. Данный инструмент пользуется популярностью среди исследователей в различных областях деятельности, таких как техника, медицина, экология, образование и экономика [66–70]. Современные средства накопления информации обеспечивают возможность получения больших объемов данных, что порождает вопрос о выборе

переменных для включения в модель. При наличии большого количества признаков регрессия может включать незначимые переменные, что может привести к нестабильности решения, его сильной зависимости от входных данных и к переобучению. Также возрастает вероятность наличия сильной корреляции между признаками (мультиколлинеарность), что затрудняет построение обратной матрицы из-за ее плохой обусловленности. Для отбора признаков используются фильтровые, обвертковые и встроенные методы [71–72]. Один из эффективных и широко применяемых встроенных методов для борьбы с нестабильностью и мультиколлинеарностью – это использование регуляризации L1 и L2, которые соответствуют моделям регрессии под названиями LASSO и Ridge.

Метод LASSO [73–74] часто приводит к установлению нулевых значений коэффициентов, что позволяет выполнить отбор признаков. Таким образом, достигается баланс между двумя аспектами: точностью регрессии и размером коэффициентов. Одной из сложностей обучения такой регрессии является необходимость определения значения параметра регуляризации [75]. В библиотеках машинного обучения как правило предлагается задавать вручную набор возможных значений для параметра регуляризации, необходимого для решения задачи (например, при использовании метода LASSO в библиотеке scikit-learn на Python или при применении L1 регуляризации в библиотеке Keras для обучения нейронных сетей). Для каждого заданного значения параметра находится минимум функции потерь, чтобы определить оптимальное значение. Такой подход требует дополнительной настройки параметров, а оптимальное значение коэффициента регуляризации может и не входить в список указанных вручную значений.

3.1.1. Алгоритм обучения

Уравнение линейной регрессии представляется следующим образом:

$$y = \beta_0 + \sum_{j=1}^{p} x_j \beta_j + \eta,$$

где y – зависимая переменная;

β – вектор коэффициентов регрессии размера p+1;

x – объясняющая переменная;

η – случайный остаток;
p – число признаков.

Тогда задача определения параметров регрессии с помощью метода наименьших квадратов имеет вид:

$$f(\beta) = \sum_{i=1}^{n} \left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \to \min,$$

где x_{ij} – значение j-го признака в i-м наблюдении;
y_i – i-е значение зависимой переменной; n – число наблюдений.

Метод LASSO представляет собой L1-регуляризацию линейной регрессии. Таким образом, в процессе настройки параметров модели осуществляется одновременная минимизация не только стандартной функции потерь, которая выражается как сумма квадратов ошибок, но и суммы абсолютных значений коэффициентов. В математическом виде задача оптимизации имеет вид

$$\sum_{i=1}^{n} \left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \left| \beta_j \right| \to \min, \tag{3.1}$$

где λ – параметр регуляризации.

Предложенный алгоритм основан на реформулировании LASSO (3.1) в виде обратной задачи при минимизации суммы абсолютных значений коэффициентов:

$$g(\beta) = \sum_{j=0}^{p} |\beta_j| \to \min,$$

$$f(\beta) = \sum_{i=1}^{n} \left(y_i - \widehat{y}_i \right)^2 = y^*, \tag{3.2}$$

где $\widehat{y}_i$ – предсказанное i-е значение; y^* – целевое значение.

Задача (3.2) в представленном ранее алгоритме (п.1.3.1) решается итерационно до выполнения условия останова, которым в том числе служит факт достижения функцией f значения y^*. В случае обучения LASSO-регрессии считается, что целевое значение y^* является некоторым заранее неизвестным малым

числом, которое не будет достигнуто, поэтому данное условие останова было исключено из алгоритма.

Таким образом, алгоритм обучения регрессии LASSO (Lasso-I) включает в себя пять шагов, описанных ниже (начальное значение номера итерации равно 1) [76–78]. Признак u представляет потенциальную возможность использования каждого аргумента в вычислениях, и изначально предполагается равным 1 для всех аргументов.

Шаг 1. Рассчитать $f(\beta)$, используя начальные значения параметров.

Шаг 2. Вычислить значения c_i как частные производные функции ограничения ($c_i = \frac{\partial f}{\partial \beta_i}$) и определить номер k, для которого абсолютное значение c_k является максимальным ($c_k = \max_i \left\{ u_i \cdot \left| c_i \right| \right\}$). Новое значение k-го аргумента определяется путем перемещения в сторону антиградиента:

$$\beta_k^* = \beta_k - \alpha \cdot c_k .$$

Шаг 3. Определить новое значение f, используя вычисленное значение β_k^*.

Шаг 4. Если $f_{new} > f(\beta)$, то $u_k = 0$; иначе, $\beta_k = \beta^*_k$.

Шаг 5. Проверить условие завершения: если $\sum_{j=1}^{p} u_j = 0$ или количество итераций превышает максимальное значение, алгоритм завершается. В противном случае количество итераций увеличивается на единицу, и процесс переходит к шагу 1.

Алгоритм работает по принципу координатного спуска, при котором для корректировки на каждой итерации выбирается только одна переменная. Важно отметить, что при применении предложенного метода входные характеристики должны быть нормализованы; в противном случае процесс обучения может занять много времени. Кроме того, параметр α должен быть настроен для обеспечения наилучшего решения.

3.1.2. Гибридный алгоритм обучения

Гибридный алгоритм основан на сочетании встроенного метода Lasso-I и обвёртки. Начальная фаза алгоритма включает в себя вычисление градиента функции $f(\beta)$ с последующей идентификацией коэффициента с максимальным абсолютным значением частной производной. Соответствующий параметр β_k выбирается в качестве кандидата для модификации. Когда β_k =0, целесообразность включения k-го признака определяется путем сравнительного анализа информационных критериев (IC) для альтернативных спецификаций модели: с включением и без включения исследуемого параметра. Включение нового признака выполняется исключительно при улучшении критерия; в противном случае признак исключается из пространства параметров.

Гибридный алгоритм (Lasso-H) включает в себя следующие шаги.

Шаг 1. Рассчитать $f(\beta)$, используя начальные значения параметров.

Шаг 2. Вычислить значения c_i как частные производные функции ограничения ($c_i = \dfrac{\partial f}{\partial \beta_i}$) и определить номер k, для которого абсолютное значение c_k является максимальным ($c_k = \max\limits_i \left\{ u_i \cdot \left| c_i \right| \right\}$). Новое значение k-го аргумента определяется путем перемещения в сторону антиградиента:

$$\beta_k^* = \beta_k - \alpha \cdot c_k .$$

Если $\beta_k = 0$, тогда проверить: если $IC(\beta^*) > IC(\beta)$, тогда u_k=0, иначе $\beta_k = \beta_k^*$.

Здесь IC – информационный критерий, меньшее значение которого соответствует лучшей модели. Для его оценки строится модель с уже включенными в нее признаками, для которых β не равно нулю.

Иначе (если $\beta_k \neq 0$), тогда проверить: если $f(\beta^*) > f(\beta)$, тогда u_k=0, иначе $\beta_k = \beta_k^*$.

Шаг 3. Проверить условие завершения: если $\sum_{j=1}^{p} u_j = 0$ или количество итераций превышает максимальное значение, алгоритм завершается. В противном случае количество итераций увеличивается на единицу, и процесс переходит к шагу 1.

3.1.3. Результаты экспериментов

Для проведения экспериментов использовались наборы данных из репозитория машинного обучения KEEL и UCI Machine learning Repository. В случаях, когда данные не были предварительно разделены, такое разделение выполнялось с использованием 20% наблюдений для тестовой выборки. Для ускорения процесса обучения нормализация входных данных и выходной величины проводилась с помощью z-нормализации. Для разработанных алгоритмов параметр α варьировался от 0,1 до 10^9 в геометрической прогрессии с шагом 10, а наилучший результат выбирался по значению критерия Акаике (AIC). Максимальное количество итераций алгоритма было установлено равным 10 000.

В первой части экспериментов была рассмотрена линейная регрессия. Реализация разработанных алгоритмов выполнена на языке Python 3. Для сравнения результатов алгоритмов Lasso-I и Lasso-H с существующими методами были использованы алгоритмы из библиотеки sklearn для Python:

– LASSO с кросс-валидацией (LassoCV);

– LARS с информационным критерием AIC (LassoLarsIC);

– Линейная регрессия (LinearRegression).

Дополнительно была реализована пошаговая регрессия с включением признаков и критерием AIC для оценки. В гибридном алгоритме Lasso-H критерий AIC также использовался для отбора признаков.

Результаты моделирования представлены в таблицах 3.1–3.3. Здесь p^* – количество признаков в исходном наборе, p – количество признаков, отобранных алгоритмами, MSE – среднеквадратичная

ошибка. AIC использовался в качестве основного инструмента для сравнения моделей.

Согласно полученным результатам, применение линейной регрессии практически во всех рассмотренных случаях приводило к высоким значениям коэффициента инфляции дисперсии (VIF), что свидетельствует о наличии мультиколлинеарности. Использование методов отбора признаков позволило снизить значения VIF и в большинстве случаев устранить проблему мультиколлинеарности.

Пошаговая регрессия продемонстрировала высокую эффективность на выборках среднего размера (от 10 до 200 признаков), обеспечив минимальное число отобранных признаков и наилучшие значения AIC благодаря оптимальному соотношению между сложностью модели и качеством её подгонки. При существенном увеличении числа признаков данный метод становится вычислительно затратным: в частности, при работе с набором данных UJIIndoorLoc его применение приводило к ошибкам времени выполнения. Алгоритм Lasso-H обеспечил наилучшие значения AIC в 46% случаев, причём в большинстве из них оптимальные решения совпадали с решениями, полученными методом пошаговой регрессии. Метод Lasso-I, в свою очередь, демонстрировал более консервативную стратегию отбора, стабильно включая в итоговые модели большее число признаков. Примечательно, что, невзирая на различия в количестве отобранных предикторов, коэффициент детерминации (R^2) оставался практически неизменным для всех методов в рамках одного набора данных: например, на наборе abalone значение R^2 оставалось на уровне 0,54 вне зависимости от применяемого алгоритма отбора. Данная закономерность свидетельствует о том, что качество подгонки модели сохраняется даже при значительном сокращении пространства признаков.

Прогностический потенциал моделей исследовался посредством вычисления MSE и коэффициента детерминации R^2 на тестовой выборке; соответствующие результаты представлены в таблице 3.4

Таким образом, полученные результаты свидетельствуют о том, что отбор признаков способен оказывать положительное воздействие на качество модели, снижать мультиколлинеарность и сложность модели, а также повышать точность прогнозирования на новых данных. Разработанные алгоритмы Lasso-I и Lasso-H, в свою очередь, продемонстрировали способность конкурировать с существующими методами, обеспечивая превосходящие результаты на отдельных наборах данных по рассматриваемым метрикам качества.

Таблица 3.1

Результаты моделирования при использовании LinearRegression и LassoCV

Датасет	p^*	LinearRegression					LassoCV				
		p	MSE	AIC	VIF	$\hat{R}^2$	p	MSE	AIC	VIF	$\hat{R}^2$
abalone	8	8	4,78	14727,99	103	0,54	8	4,78	14727,99	102	0,54
ailerons	40	40	$3{,}01 \cdot 10^{-8}$	-159225,6	∞	0,82	40	$3{,}01 \cdot 10^{-8}$	-159208,73	∞	0,82
autoMPG8	7	7	10,56	1642,13	23	0,83	6	10,64	1644,5	13	0,82
baseball	16	16	465543,09	4308,1	21	0,68	16	465556,10	4308,1	21	0,68
compactiv	21	20	81013451755	183236,08	18	0,54	20	81013451856	183236,08	18	0,54
house	16	16	2103049809	443032,7	77	0,26	16	2103055262	443032,74	77	0,26
Mv	10	10	20,20	190604,92	2	0,81	9	20,20	190605,62	2	0,81
puma32h	32	32	0,000721	-28744,05	1	0,22	2	0,000725	-28710,94	1	0,22
wankara	9	9	2,42	973,01	52	0,99	7	2,45	975,99	11	0,99
wizmir	9	9	1,53	3832,2	14	0,99	9	1,53	3832,4	14	0,99
communities and crime	100	100	0,0166	-1803,99	1404	0,681	23	0,0187	-1616,39	21	0,659
Large-scale Wave Energy Farm	98	98	2527594013	706296,84	16	0,83	98	2527594015	706296,84	16	0,83
UJIIndoorLoc	465	465	856,31	192137,87	29885	0,94	236	883,25	192755,56	1	0,94
BlogFeedback	276	276	903,19	505860,69	∞	0,36	31	911,36	506332,49	∞	0,36

Таблица 3.2

Результаты моделирования при использовании Stepwise regression и LassoLarsIC

Датасет	p^*	Stepwise regression					LassoLarsIC				
		p	MSE	AIC	VIF	$\hat{R}^2$	p	MSE	AIC	VIF	$\hat{R}^2$
abalone	8	7	4,78	**14726,05**	103	0,54	7	4,78	14726,07	102	0,54
ailerons	40	14	$3{,}01 \cdot 10^{-8}$	**-159251,62**	144	0,82	11	$3{,}11 \cdot 10^{-8}$	-158917,71	80	0,81
autoMPG8	7	5	10,62	**1639,78**	12	0,83	6	10,62	1641,65	13	0,83
baseball	16	9	476094,09	**4300,13**	6	0,68	15	467380,03	4307,16	14	0,68
compactiv	21	19	81013455763	**183234,08**	17	0,54	19	81014583097	183234,17	17	0,54
house	16	13	2104427110	443038,63	45	0,26	16	2103049809	443032,7	77	0,26
Mv	10	6	20,20	**190598,64**	2	0,81	9	20,20	190602,92	2	0,81
puma32h	32	5	0,000723	**-28781,04**	1	0,22	2	0,000725	–28773,69	1	0,22
wankara	9	6	2,43	**968,01**	51	0,99	6	2,46	970,51	11	0,99
wizmir	9	6	1,53	**3826,86**	14	0,99	9	1,53	3832,2	14	0,99
communities and crime	100	39	0,0174	-1858,61	40	0,680	80	0,0168	-1825,27	113	0,681
Large-scale Wave Energy Farm	98	92	2527928369	**706288,65**	15	0,83	98	2527594013	706296,84	16	0,83
UJIIndoorLoc	465	–	–	**–**	–	–	345	861,30	**192013,86**	11	0,94
BlogFeedback	276	84	904,87	**505574,31**	∞	0,36	19	912,98	505911,67	∞	0,36

Таблица 3.3

Результаты моделирования при использовании Lasso-I и Lasso-H

Датасет	p^*	Lasso-I					Lasso-H				
		p	MSE	AIC	VIF	*R2*	*p*	MSE	AIC	VIF	*R2*
abalone	8	7	4,78	**14726,05**	102	0,54	7	4,78	**14726,05**	102	0,54
ailerons	40	10	$3{,}02 \cdot 10^{-8}$	-159218,05	177	0,82	9	$3{,}02 \cdot 10^{-8}$	-159219,35	154	0,82
autoMPG8	7	7	10,56	1642,13	23	0,83	3	10,77	1640,07	2	0,82
baseball	16	16	465543,09	4308,10	21	0,68	8	482044,17	4301,47	5	0,68
compactiv	21	20	81013451755	183236,08	18	0,54	19	81013455763	**183234,08**	17	0,54
house	16	15	2103060558	**443030,79**	77	0,26	15	2103060558	**443030,79**	77	0,26
mv	10	10	20,20	190604,92	2	0,81	6	20,20	**190598,64**	2	0,81
puma32h	32	1	0,000725	-28774,96	1	0,22	1	0,000725	-28774,96	1	0,22
wankara	9	5	2,45	968,33	9	0,99	5	2,45	968,33	9	0,99
wizmir	9	7	1,53	3828,42	14	0,99	6	1,53	**3826,86**	14	0,99
communities and crime	100	89	0,0167	-1825,19	1369	0,683	38	0,0174	**-1862,85**	39	0,680
Large-scale Wave Energy Farm	98	98	2527594013	706294,84	16	0,83	92	2528818170	706296,8	15	0,83
UJIIndoorLoc	465	137	928,98	193103,95	8	0,94	136	928,98	193101,97	8	0,94
BlogFeedback	276	208	903,66	505749,70	∞	0,36	81	905,15	505582,34	∞	0,36

Таблица 3.4

Результаты моделирования на тестовой выборке

Датасет	LinearRegression		LassoCV		LassoLarsIC		Stepwise regression		Lasso-I		Lasso-H	
	MSE	R^2	MSE	R^2	MSE	R^2	MSE	R^2	MSE	R^2	MSE	R^2
abalone	5,31	0,48	5,31	0,48	5,30	0,48	5,31	0,48	5,31	0,48	5,31	0,48
ailerons	$3{,}04 \cdot 10^{-8}$	0,82	$3{,}02 \cdot 10^{-8}$	0,83	$3{,}05 \cdot 10^{-8}$	0,82	$3{,}03 \cdot 10^{-8}$	0,83	$3{,}02 \cdot 10^{-8}$	0,83	$3{,}02 \cdot 10^{-8}$	0,83
autoMPG8	12,15	0,78	12,18	0,77	12,21	0,78	12,37	0,78	12,15	0,78	12,51	0,77
baseball	456585,30	0,7	455427,51	0,68	446174,61	0,69	486258,51	0,67	456585,30	0,69	473899,89	0,68
compactiv	78859918598	0,56	78860020529	0,56	78871798063	0,56	78860165749	0,56	78859918598	0,56	78860165749	0,56
house	1907341796	0,25	1907197674	0,25	1907341796	0,25	1910312382	0,25	1907357317	0,25	1907357317	0,25
Mv	19,98	0,82	19,97	0,82	19,98	0,82	19,97	0,82	19,98	0,82	19,97	0,82
puma32h	0,000677	0,22	0,000673	0,22	0,000673	0,22	0,000675	0,22	0,000673	0,22	0,000673	0,22
wankara	**2,75**	0,99	2,82	0,99	2,82	0,99	2,80	0,99	2,92	0,99	2,92	0,99
wizmir	1,73	0,99	1,73	0,99	1,73	0,99	1,73	0,99	1,73	0,99	1,73	0,99
communities and crime	0,0173	0,64	0,0173	0,64	0,0172	0,64	0,0179	0,63	0,0174	0,64	0,0173	0,64
Large-scale Wave Energy Farm	2472748809	0,83	2472753672	0,83	2472748809	0,83	2471353990	0,83	2472748809	0,83	2472422504	0,83
UJI-Indoor-Loc	2414,17	0,84	2172,80	0,85	2285,83	0,85	–	–	2130,45	0,85	2130,22	0,85
Blog-Feedback	482,57	0,67	505,05	0,70	532,56	0,68	476,80	0,70	485,65	0,70	484,28	0,70

Разработанные алгоритмы также были применены для построения логистической регрессии (результаты представлены в таблицах 3.6–3.7). Для сравнения использовалась пошаговая регрессия и методы библиотеки sklearn: логистическая регрессия (LogisticRegression) и LASSO с кросс-валидацией (LogisticRegressionCV).

В соответствии с полученными результатами, разработанный алгоритм Lasso-H продемонстрировал наименьшие значения AIC на обучающей выборке в 90% случаев, что подтверждает его способность строить более сбалансированные модели по сравнению с альтернативными методами. В частности, алгоритм Lasso-H показал более низкие значения AIC на наборах данных magic (13986,57), ionosphere (126,77) и wdbc (72,83) относительно всех рассматриваемых алгоритмов, включая LogisticRegressionCV и пошаговую регрессию.

Использование базового метода LogisticRegression привело к критически высокие значения VIF: например, на наборах данных wdbc (4180), coil2000 (1704) и appendicitis (515). Пошаговая регрессия и методы на основе Lasso-регуляризации, в свою очередь, обеспечили существенно меньшие значения VIF.

При оценке на тестовой выборке (таблица 3.8) алгоритм Lasso-H также продемонстрировал конкурентоспособные результаты, достигнув более высокого или сопоставимого качества классификации по показателям F1-меры и площади под ROC-кривой (AUC) на таких наборах данных, как ionosphere (F1 = 0,903, AUC = 0,853) и magic (F1 = 0,676, AUC = 0,843). Более того, в ряде случаев – в частности, на наборе данных wdbc – рассматриваемый алгоритм превосходил прочие методы по точности классификации (Acc) либо не уступал им, сохраняя при этом меньшую сложность результирующей модели.

Таблица 3.6

Результаты моделирования логистической регрессии с использованием методов LogisticRegression, LogisticRegressionCV и пошаговой регрессии

Датасет	p^*	LogisticRegression				LogisticRegressionCV				Stepwise regression			
		p	AIC	BIC	VIF	p	AIC	BIC	VIF	p	AIC	BIC	VIF
appendicitis	7	7	70,97	90,41	515	5	56,68	71,27	39	3	66,61	76,33	36
coil2000	85	85	3225,00	3824,35	1704	54	3196,32	3572,66	1694	23	**3152,07**	3319,33	5
ionosphere	33	33	192,90	316,48	8	21	146,40	226,36	5	11	164,85	208,46	3
magic	10	10	14001,86	14085,80	28	10	13992,01	14075,94	28	5	14436,55	14482,33	3
monk-2	6	6	289,61	316,52	1	5	289,03	312,09	1	2	282,05	293,58	1
phoneme	5	5	4100,22	4138,45	1	5	4122,06	4160,29	1	5	4100,22	4138,45	1
pima	8	8	617,80	657,58	2	8	603,09	642,87	2	2	748,64	761,90	1
spambase	57	57	1590,20	1950,37	126	55	1560,70	1908,45	3	41	1706,67	1967,49	2
spectfheart	44	44	212,54	363,80	14	1	234,38	237,74	–	0	224,31	227,67	–
twonorm	20	20	700,38	840,79	1	20	1181,73	1315,45	1	20	700,38	840,79	1
wdbc	30	30	149,43	277,16	4180	21	82,16	172,80	3754	5	299,38	324,10	20

Таблица 3.7

Результаты моделирования логистической регрессии с использованием методов Lasso-I и Lasso-H

Датасет	p*	Lasso-I				Lasso-H			
		p	AIC	BIC	VIF	*p*	AIC	BIC	VIF
appendicitis	7	5	58,87	73,45	158	3	**53,48**	63,20	1
coil2000	85	47	3239,18	3573,70	14	18	3162,04	3294,45	2
ionosphere	33	27	144,49	246,26	7	14	**126,77**	181,29	3
magic	10	1	20877,42	20885,05	–	7	**13986,57**	14047,61	5
monk-2	6	4	270,11	289,32	1	2	**264,58**	272,26	1
phoneme	5	5	4116,78	4155,01	1	5	**4100,20**	4138,43	1
pima	8	8	603,83	643,61	2	4	**598,74**	620,84	1
spambase	57	54	1764,31	2105,85	3	45	**1516,86**	1802,52	2
spectfheart	44	34	208,27	325,92	12	3	**170,48**	183,93	3
twonorm	20	20	709,49	849,90	1	20	**698,19**	831,92	1
wdbc	30	24	93,66	196,67	205	8	**72,83**	109,92	111

Таблица 3.8

Результаты решения задачи классификации на тестовой выборке

Датасет	LogisticRegression			LogisticRegressionCV			Stepwise regression			Lasso-I			Lasso-H		
	Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC	Acc	F1	AUC
appendicitis	0,773	0,286	0,781	0,773	0,545	0,750	0,773	0,286	0,750	0,773	0,545	0,760	0,727	0,400	0,760
coil2000	0,937	0,016	0,713	0,938	0,016	0,716	0,937	0,016	0,721	0,939	0,000	0,716	0,938	0,016	0,719
ionosphere	0,845	0,887	0,824	0,859	0,896	0,866	0,789	0,845	0,831	0,859	0,896	0,878	0,873	0,903	0,853
magic	0,791	0,671	0,842	0,794	0,674	0,843	0,785	0,658	0,831	0,737	0,649	0,788	0,794	0,676	0,843
monk-2	0,747	0,766	0,868	0,736	0,758	0,881	0,747	0,761	0,885	0,747	0,761	0,885	0,747	0,761	0,885
phoneme	0,756	0,522	0,819	0,757	0,519	0,819	0,756	0,522	0,819	0,755	0,523	0,819	0,756	0,522	0,819
pima	0,786	0,660	0,877	0,799	0,687	0,873	0,701	0,410	0,670	0,792	0,680	0,871	0,773	0,639	0,860
spambase	0,923	0,906	0,970	0,920	0,902	0,971	0,908	0,888	0,969	0,914	0,896	0,970	0,916	0,899	0,969
spectfheart	0,741	0,851	0,778	0,833	0,909	0,310	0,833	0,909	0,500	0,796	0,882	0,899	0,833	0,905	0,770
twonorm	0,980	0,980	0,998	0,980	0,981	0,998	0,980	0,980	0,998	0,978	0,979	0,998	0,980	0,980	0,998
wdbc	0,956	0,940	0,998	0,982	0,977	0,996	0,947	0,927	0,957	0,965	0,953	0,997	0,982	0,977	0,999

3.2. Обучение полносвязной нейронной сети

3.2.1. Алгоритм обучения

Рассмотренный в п. 3.1.1 алгоритм был модифицирован для обучения нейронных сетей с L1 регуляризацией. В переформулированной задаче сумма абсолютных значений параметров минимизируется, в то время как ошибка обучения ограничивается условным значением (как в регрессии LASSO) [79]:

$$\|w\|_1 \to \min,$$

$$J(w) = J^*,$$

где J^* – условное значение функции-ограничения.

Входные данные алгоритма:

- X – матрица входных значений (включая смещение);
- y – выходные значения;
- input_neurons – количество входных нейронов;
- hidden_neurons – количество нейронов в скрытом слое;
- u – вектор активности размерности размер($w^{(0)}$) + размер($w^{(1)}$), инициализируемый единицами.

Выходные данные:

- $w^{(0)}$ – веса первого слоя (вход → скрытый слой);
- $w^{(1)}$ – веса второго слоя (скрытый слой → выходной слой).

Начальные значения $w^{(0)}$ и $w^{(1)}$ инициализируются нулевыми значениями: $w^{(0)}$ = нулевая_матрица(input_neurons+1, hidden_neurons), $w^{(1)}$ = нулевой_вектор(hidden_neurons+1).

Алгоритм обучения нейронной сети с одним скрытым слоем с L1 регуляризацией (Sparse) будет включать следующие шаги:

Шаг 1. Вычислить значение функции потерь $J_{prev}=J(w)$ (2.2).

Шаг 2. Вычислить градиенты по формулам (2.3) и (2.4):

$$g_0 = X^T l_{delta},$$

$$g_1 = l_{bias}{}^T y_{delta}.$$

Шаг 3. Определить максимальное значение произведения градиента на признак возможности использования в расчётах:

$$g_{\max} = \max\left\{|g_0| \cdot u_0, |g_1| \cdot u_1\right\}.$$

Если указанное максимальное значение соответствует весовым коэффициентам, связывающим скрытый слой с выходным (v – индекс максимального элемента), производится корректировка весового коэффициента $w^{(1)}$:

$$w_v^{*(1)} = w_v^{*(1)} - \eta \cdot g_{1v},$$

где η – параметр, определяющий шаг изменения весового коэффициента.

В противном случае обновляется весовой коэффициент, связывающий входной слой со скрытым (b, v – индексы максимального элемента):

$$w_{b,v}^{*(0)} = w_{b,v}^{*(0)} - \eta \cdot g_{0b,v}.$$

Шаг 4. Вычисление нового значения функции ошибки $J_{new}=J(w^*)$ (2.2).

Если $J_{new}<J_{prev}$, значения u для всех весовых коэффициентов полагаются равными 1, $w=w^*$, $J_{prev}= J_{new}$. Переход к шагу 2.

В противном случае значение u для соответствующего скорректированного весового коэффициента устанавливается равным нулю: $u_{0_{b,v}} = 0$ (если $w_{b,v}^{(0)}$ был изменен) или $u_{1_v} = 0$ (если $w_v^{(1)}$ был изменен).

Переход к шагу 2.

Критерий останова: обнуление всех значений u либо выполнение заданного числа итераций.

3.2.2. Результаты экспериментов

Для сравнения результатов работы алгоритма обучения нейронной сети с L1 регуляризацией был рассмотрен алгоритм Adam, реализованный в библиотеке Keras, с различными параметрами регуляризации L1 и методами отсева. Для настройки гиперпараметров был использован алгоритм GridSearchCV из библиотеки scikit-learn, который выполняет поиск по указанным значениям параметров, оценивая каждую комбинацию с помощью перекрестной проверки для определения оптимальных гиперпараметров, включая количество периодов обучения и размер пакета.

Эксперименты были проведены в рамках двух сценариев: в первом сценарии рассматривалась однослойная сеть, а во втором – нейронная сеть с одним скрытым слоем.

В первом сценарии были проведены эксперименты с использованием данных финансовых временных рядов рыночного индекса Wilshire 2500. Экспериментальная установка включала прогнозирование будущих значений индекса на основе 40-дневного исторического периода с использованием ежедневных данных с января 2021 по декабрь 2023 года (всего 754 наблюдения, последние 200 наблюдений были применены для тестирования).

Результаты, включая MSE и MAE для тестового набора, представлены в таблице 3.9. В таблице 3.9 также указано значение параметра регуляризации λ, при этом arctg используется в качестве функции активации однослойной нейронной сети. Предложенный алгоритм продемонстрировал эффективность по сравнению с традиционными подходами, в то время как классической регуляризацией L1 (λ = 0,01–1,0) отбор признаков не был выполнен – все 40 входных признаков были сохранены при уменьшении значений коэффициентов. При этом предложенный алгоритм идентифицировал только 6 признаков, которые обеспечили лучшую точность прогнозирования.

Результаты, приведенные в таблице 3.9, позволяют сделать следующие выводы. Стандартная оптимизация Adam без регуляризации позволила достичь значения MSE = $4{,}39 \cdot 10^{-5}$, в то время как предложенный алгоритм снизил это значение до значения MSE = $2{,}48 \cdot 10^{-6}$. Кроме того, была повышена эффективность вычислений, что позволило сократить время обучения с 11,3 секунды до 0,43 секунды. Это улучшение связано со способностью алгоритма устранять ненужные параметры во время обучения, а не штрафовать за них.

Таблица 3.9

Результаты вычислительных экспериментов на основе индекса Wilshire 2500

Метод	Число отобранных признаков, *p*	MSE	MAE	Время, сек.
Adam, λ=0	40	$4{,}39 \cdot 10^{-5}$	0,0053	11,3
Adam, λ=0,01	40	$2{,}03 \cdot 10^{-4}$	0,0103	12,8
Adam, λ=0,5	40	0,0023	0,0105	12,8
Adam, λ=1,0	40	0,0048	0,0101	7,45
Sparse, η=0,1	6	$2{,}48 \cdot 10^{-6}$	0,0013	0,4347

На рисунке 3.4 показаны фактические и прогнозируемые с помощью предложенного алгоритма значения на тестовой выборке. Анализ различных функций активации (рисунок 3.5) показал, что алгоритм Sparse обеспечил более низкую величину ошибки, чем традиционный метод. Количество выбранных признаков с использованием Sparse варьировалось от 2 до 10 для различных функций активации.

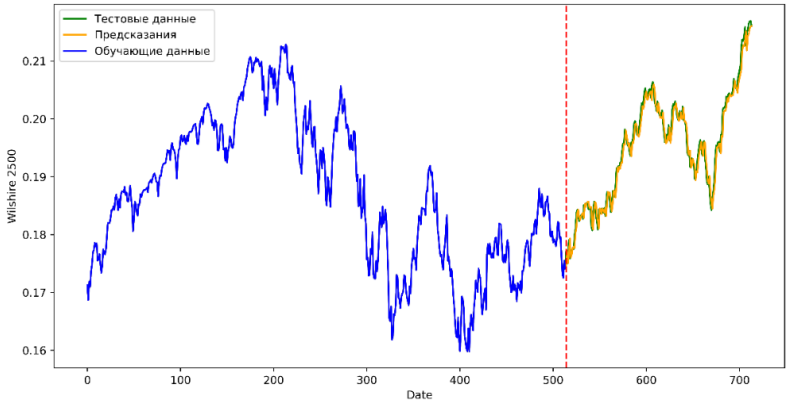


Рис. 3.4. Фактические и прогнозные значения индекса

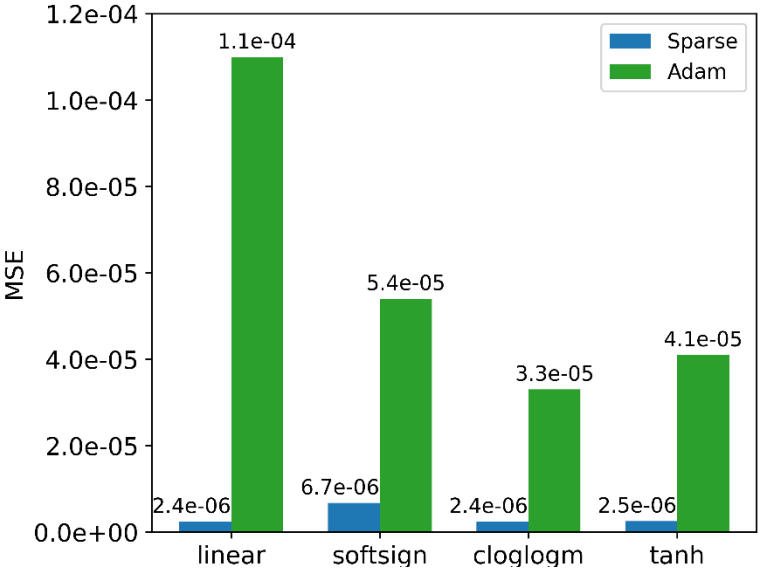


Рис. 3.5. Значения MSE в зависимости от функции активации

Во втором сценарии экспериментов была оценена производительность алгоритма Sparse на архитектуре с одним скрытым слоем [41] с использованием финансовых данных предприятия. Был использован набор данных о 551 российских предприятиях с выручкой более 100 млн. рублей за 2017–2020 годы [36]. Обучающая выборка включала информацию об 551 предприятии за 2017–2019 годы, тестовая выборка включала данные по тем же предприятиям за 2020 год. Рассматривалась задача прогнозирования рентабельности предприятия на основе финансовых показателей (коэффициент ликвидности, доля основных средств, финансовый рычаг и оборачиваемость активов). В ходе экспериментов сравнивались сети с 2 и 16 нейронами в скрытом слое. Для скрытого слоя была использована сигмоидальная функция активации, а для выходного – линейная.

Результаты, приведенные в таблице 3.10, показывают, что для сети из 2 нейронов предложенный алгоритм уменьшил число параметров с 13 до 8, улучшив MSE с 174,79 до 156,28 и MAE с 8,54 до

8,06 по сравнению полносвязной сетью и оптимизатором Adam. Для сети с 16 нейронами в скрытом слое алгоритм добился существенного сокращения параметров с 97 до 13 при сохранении сопоставимой точности (MSE = 155,01 против 156,00), продемонстрировав эффективное упрощение модели без снижения производительности.

Кроме того, алгоритм продемонстрировал превосходство над dropout – методом регуляризации нейронных сетей, который предотвращает переобучение путем случайного "выключения" (обнуления) части нейронов во время обучения. В то время как dropout с вероятностью отсева 20% обеспечил MSE = 195,88, предложенный алгоритм достиг значения MSE = 155,01 при существенном снижении числа параметров, что указывает на более эффективную регуляризацию за счет выбора параметра, а не случайного отсева. Время обучения алгоритма Sparse составило 9,1 секунды, тогда как Adam с dropout потребовал 32,8 секунды.

Таблица 3.10

Результаты моделирования нейронной сети со скрытым слоем

Метод	Число нейронов в скрытом слое	Число ненулевых параметров	MSE	MAE	Время, сек.
Adam, λ=0	2	13	174,79	8,54	33,2
Adam, λ=0,01	2	13	176,54	8,54	29,9
Adam, λ=0,5	2	13	182,19	8,54	29,2
Adam, λ=1,0	2	13	196,53	8,89	28,5
Предложенный алгоритм, η=0,0001	2	8	156,28	**8,06**	0,6
Adam, λ=0	16	97	156	8,08	25,8
Adam, λ=0,01	16	97	156,99	8,09	22,2
Adam, λ=0,5	16	97	178,7	8,34	22,9
Adam, λ=1,0	16	97	193,43	8,6	23,7
Adam, dropout (0,2)	16	97	195,88	8,7	32,8
Sparse, η=0,0001	16	13	**155,01**	8,08	9,1

Рисунок 3.6 иллюстрирует чувствительность алгоритма к параметру η, демонстрируя оптимальную производительность при $\eta = 10^{-4}$ в рассмотренной задаче. При $\eta=10^{-2}$ выбор параметров не был выполнен, поскольку оптимизированная функция принимала более высокие значения при значительном изменении аргументов по сравнению с их нулевыми значениями. При $\eta=10^{-6}$ наблюдалось медленное снижение оптимизированной функции из-за использования небольшого шага.

Эксперименты показывают, что алгоритм применим для обучения разреженных нейронных сетей, где скорость обучения и интерпретируемость модели важны не меньше, чем точность предсказаний.

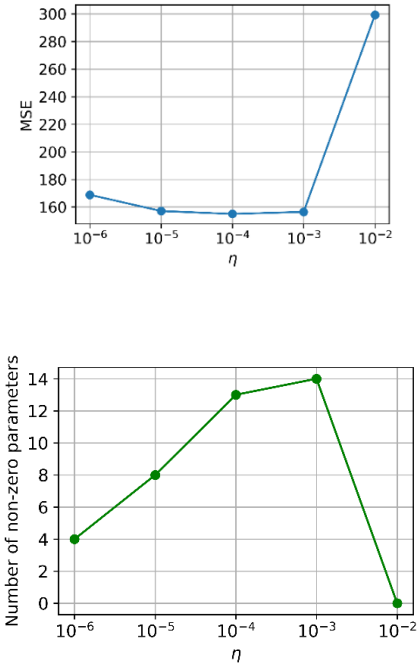


Рис. 3.6. Изменение MSE и числа ненулевых параметров при изменении η

3.3. Гибридное обучение нейронной сети с отбором признаков

Для обучения нейронной сети может быть рассмотрено применение гибридных оптимизационных алгоритмов, объединяющих методы различной природы для использования сильных стороны каждого из них. В обширном обзоре Kaveh и Mesgari [80], охватывающем свыше 627 публикаций, показано, что гибриды метаэвристик с градиентными методами в среднем обеспечивают лучший компромисс между скоростью сходимости и качеством найденного решения по сравнению с однородными подходами. Вместе с тем авторы констатируют, что сложность проектирования и настройки подобных архитектур существенно ограничивает публикационную активность в данном направлении.

Анализ литературы свидетельствует о том, что большинство работ, посвящённых гибридному обучению, сосредоточено на комбинировании методов в рамках единого алгоритма, применяемого ко всей сети без дифференциации по слоям. Авторы применяют два основных подхода к гибридизации: встраивание популяционных метаэвристик (рой сальп, рой частиц, серые волки) в качестве глобального оптимизатора с последующим локальным уточнением градиентными методами [81–82], а также модификацию самого метаэвристического алгоритма специализированными операторами для ускоренного обучения нейросетей [83].

Предлагаемый алгоритм основан на идее послойной гибридизации: входной слой настраивается посредством L1-регуляризации (метод Sparse), которая формирует разреженную структуру весов и тем самым осуществляет отбор признаков непосредственно в ходе обучения; для скрытого слоя применяется стандартный метод Adam (рисунок 3.7).

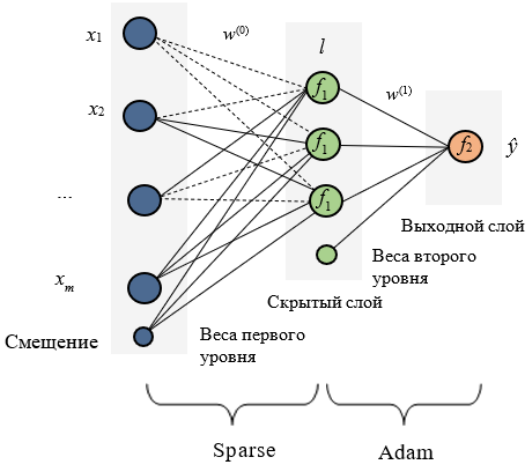


Рис. 3.7. Модель нейронной сети с отбором признаков в первом слое: x – входные значения, $\hat{y}$ – выходные значения, w – настраиваемые в ходе обучения весовые коэффициенты, l – выход скрытого слоя (пунктиром обозначены весовые коэффициенты с нулевыми значениями)

3.3.1. Алгоритм обучения

Гибридный алгоритм обучения (Hybrid) рассмотрен на примере сети с одним скрытым слоем (рисунок 2.4) и основан на следующих правилах:

- Первый слой настраивается с использованием метода минимизации суммы абсолютных значений весовых коэффициентов (Sparse) [13, 14], что обеспечивает отбор значимых весовых коэффициентов и признаков.
- Второй слой оптимизируется с использованием стандартного оптимизатора Adam.

Входные данные алгоритма:

- максимальное число итераций T;
- X – матрица входных значений (включая смещение);
- y – выходные значения;
- input_neurons – количество входных нейронов;

- hidden_neurons – количество нейронов в скрытом слое;
- α – параметр шага обновления весов α;
- u – вектор активности размерности размер($w^{(0)}$) + размер($w^{(1)}$), инициализируемый единицами.

Выходные данные:

- $w^{(0)}$ – веса первого слоя (вход → скрытый слой);
- $w^{(1)}$ – веса второго слоя (скрытый слой → выходной слой).

Начальные значения $w^{(0)}$ полагаются равными нулю; веса $w^{(1)}$ инициализируются случайными величинами с нормальным распределением: $w^{(0)}$ = нулевая_матрица(input_neurons+1, hidden_neurons), $w^{(1)}$ = случайный_вектор(hidden_neurons+1). Начальные значения переменных: $l = f_1(X \cdot w^{(0)})$, $l_{bias} = [l, 1]$, $\hat{y} = l_{bias} \cdot w^{(1)}$, $J = \|y - \hat{y}\|^2$, $t = 0$ (счётчик итераций).

Шаг 1. Модификация весов первого слоя методом Sparse.

Шаг 1.1 Поиск весового коэффициента с максимальным абсолютным значением градиента:

$$\delta = (\hat{y} - y) \odot f_2'(\hat{y}) \cdot w^{(1)}[:-1]^{\mathrm{T}} \odot f_1'(X \cdot w^{(0)})$$

Расчет абсолютных значений градиента с учетом активных весов:

$$g_0 = |X^{\mathrm{T}} \cdot \delta|.$$

Определение индекса максимального элемента

$$p = \text{индекс_максимума}(g_0).$$

Индекс столбца (нейрон):

$$v = p \bmod \text{hidden_neurons}.$$

Индекс строки (вход):

$$b = p \,/\, \text{hidden_neurons}.$$

Шаг 1.2 Изменение весового коэффициента:

$$w^{(0)}_{b,v} = w^{(0)}_{b,v} - \alpha \cdot g_{0b,v}.$$

Шаг 2. Модификация весов второго слоя методом Adam:

$$t = t + 1$$

$$w^{(1)} = w^{(1)} - \alpha \cdot \frac{m_corr}{\sqrt{\gamma_corr + \varepsilon}},,$$

где m_corr – скорректированная оценка первого момента;

γ_corr – скорректированная оценка второго момента.

Если $t > T$, то завершение работы, иначе – переход на шаг 1.

3.3.2. Результаты экспериментов

Для проведения исследования были выбраны датасеты из репозиториев KEEL и UCI с числом признаков от 85 до 100 и с числом наблюдений от 1994 до 9822: Community and crime, Tic, Coil 2000, Taiwanese Bankruptcy Prediction. Первые два датасета предназначены для решения задачи регрессии, а вторые два – классификации. Для входных значений использовалась z-нормализация, а для выходной величины в задаче регрессии Min–Max нормирование. Сформированная выборка была разделена на обучающую и тестовую подвыборки в соотношении 80% к 20% соответственно.

В экспериментах использовались следующие архитектурные гиперпараметры: количество нейронов скрытого слоя принимало значения 2, 12 или 24; функция активации первого слоя f_1 выбиралась из набора {sigmoid, linear, cloglogm, snake}; темп обучения α варьировался в диапазоне [0,1; 0,01; 0,001; 0,0001], число эпох составляло 700. Для выходного нейрона была использована функция активация linear для задачи регрессии и sigmoid для задачи классификации.

В таблицах 3.11–3.12 приведены наилучшие результаты на тестовой выборке по MSE (в случае задачи классификации данная метрика называется Brier Score). Степень разреженности была определена по формуле:

$$Sparsity = 1 - \frac{non \quad zero \quad weights}{total \quad weights},$$

где *non zero weights* – число ненулевых весовых коэффициентов нейронной сети;
total weights – общее число весовых коэффициентов.

Согласно полученным результатам, гибридный алгоритм обеспечил сопоставимую точность при меньшем числе весовых коэффициентов. Так, например, для датасета Community and crime при использовании гибридного алгоритма было достигнуто чуть меньшее значение MSE на тестовой выборке по сравнению с методом Adam при разреженности 86%.

Таблица 3.11

Результаты моделирования для задачи регрессии

	Community and crime		Tic	
	Adam	Hybrid	Adam	Hybrid
MSE train	0,0168	0,0164	0,0532	0,0524
MSE test	0,0172	0,0165	0,0528	0,0525
Число нейронов в скрытом слое	24	12	12	2
Число признаков	100	71	85	38
Разреженность	0	0,86	0	0,75

Таблица 3.12

Результаты моделирования для задачи классификации

	Coil 2000		Taiwanese Bankruptcy Prediction	
	Adam	Hybrid	Adam	Hybrid
MSE train	0,0518	0,0526	0,0165	0,0226
MSE test	0,0553	0,0547	0,0258	0,0256
Число нейронов в скрытом слое	12	2	12	24
Число признаков	85	15	95	39
Разреженность	0	0,88	0	0,93

На рисунке 3.8 показано изменение средней разреженности для гибридного алгоритма в случае изменения числа нейронов в скрытом слое. Можно увидеть, что для большинства датасетов разреженность увеличивается. На рисунке 3.9 показано время обучения для двух алгоритмов, для гибридного алгоритма оно стабильно меньше.

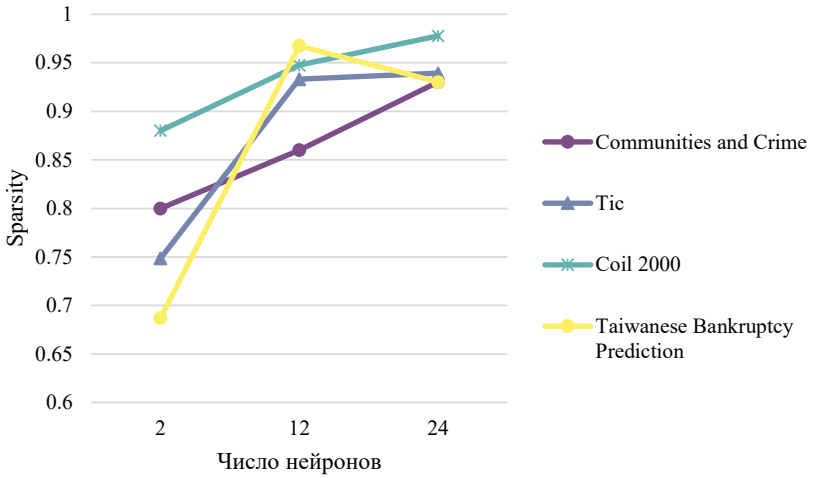


Рис. 3.8. Разреженность гибридного алгоритма

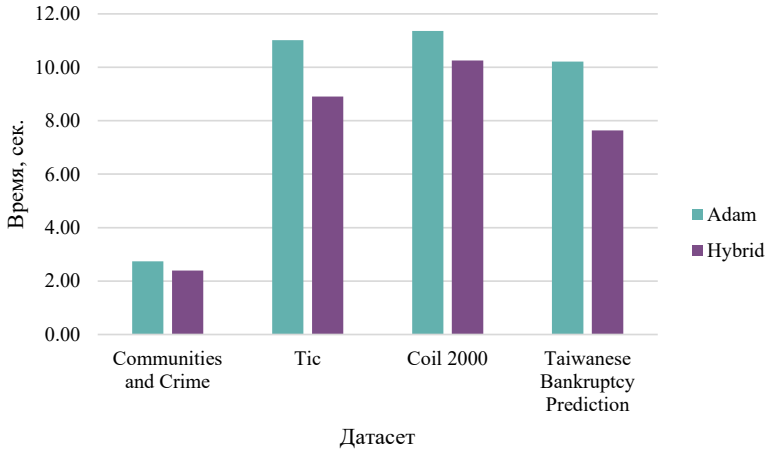


Рис. 3.9. Среднее время обучения

3.4. Гибридное обучение разреженной нейронной сети без отбора признаков

В данном разделе рассматривается алгоритм послойной гибридизации, в рамках которого первый слой нейронной сети обучается с помощью оптимизатора Adam, а последующие слои (второй и третий) оптимизируются алгоритмом формирования разреженной сети (Sparse) (рисунок 3.10) [84]. Таким образом образуется полносвязный первый слой без отбора признаков, а последующие слои формируются разреженными для устранения параметрической избыточности. Рассмотрено два гибридных алгоритма. В первом алгоритме (Adam-S-1) для разрежения используется метод Sparse; во втором алгоритме (Adam-S-2) выполнена модификация данного подхода, обусловленная зашумленностью градиентов: для выбора параметра, подлежащего изменению на каждой итерации, применяются сглаженные обновления, вычисляемые по методу Adam.

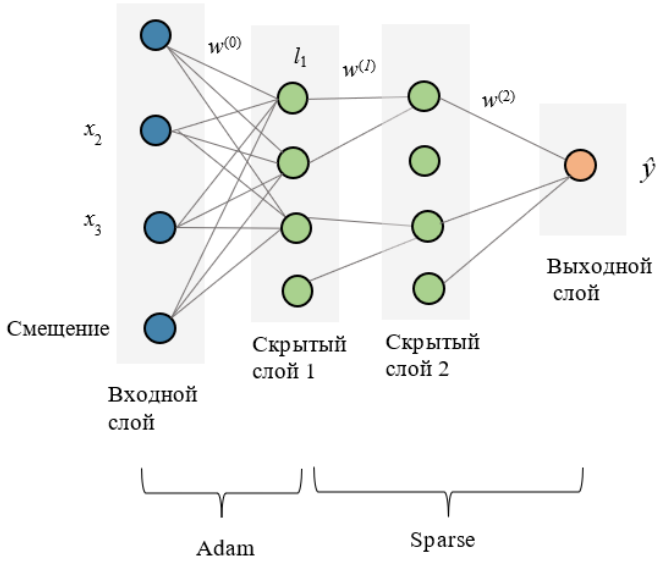


Рис. 3.10. Модель нейронной сети: x – входные значения, $\hat{y}$ – выходные значения, w – настраиваемые в ходе обучения весовые коэффициенты, l_i – выход i-го слоя

3.4.1. Алгоритм обучения

Алгоритм Adam-S-2, реализующий послойную гибридизацию, характеризуется следующими входными и выходными данными.

Входные данные:

- X – матрица входных значений (включая смещение);
- y – выходные значения;
- input_neurons – количество входных нейронов;
- hidden_neurons1 – количество нейронов в первом скрытом слое;
- hidden_neurons2 – количество нейронов во втором скрытом слое;
- α – параметр шага изменения весового коэффициента первого слоя;
- β – параметр шага изменения весового коэффициента второго и третьего слоя;
- μ_1=0,9, μ_2=0,99 – коэффициенты затухания для первого и второго моментов Adam;
- $\varepsilon=10^{-8}$ – малое число для предотвращения деления на ноль;
- T – максимальное число итераций.

Выходные данные:

- $w^{(0)}$– веса первого слоя (вход → скрытый слой 1);
- $w^{(1)}$– веса второго слоя (скрытый слой 1 → скрытый слой 2);
- $w^{(2)}$ – веса третьего слоя (скрытый слой 2 → выходной слой).

Алгоритм Adam-S-2 включает следующую последовательность шагов.

Шаг 1. Инициализация параметров сети.

Инициализировать веса: $w^{(0)}$, $w^{(1)}$ и $w^{(2)}$. Обнулить моменты Adam. Установить счетчик шагов t=0, маску активности $mask$ = вектор единиц размера N (N – общее число нейронов в сети), текущее число итераций k=0.

Шаг 2. Запуск основного цикла обучения.

Пока ($k < T$) и (сумма $mask > 0$) выполнять шаги 3–10.

По завершении цикла вернуть итоговые матрицы весов $w^{(0)}$, $w^{(1)}$, $w^{(2)}$.

Шаг 3. Прямое распространение и вычисление функции потерь.

Выполнить прямое распространение: $l_1 = f_1(X \cdot w^{(0)})$, $l_2 = f_1([l_1,1] \cdot w^{(1)})$, $\hat{y} = f_2([l_2,1] \cdot w^{(2)})$. Вычислить ошибку $J = || \hat{y} - y||^2$ (2.2).

Из общего вектора маски *mask* выделяются подвекторы *mask_w1* и *mask_w2*, соответствующие весам второго и третьего слоев, которые затем преобразуются в матрицы размерности (hidden_neurons1+1) × hidden_neurons2 и (hidden_neurons2+1) соответственно.

Шаг 4. Обратное распространение (вычисление градиентов).

Вычислить градиенты для всех слоев: w_2_grad, w_1_grad, w_0_grad через дельты ошибок.

Шаг 5. Обновление моментов Adam.

Увеличить $t = t + 1$. Обновить первые и вторые моменты для всех слоев с коэффициентами μ_1 и μ_2. Выполнить коррекцию смещения моментов.

Шаг 6. Вычисление обновлений Adam.

Рассчитать скорректированные обновления w_0_update, w_1_update, w_2_update по формуле

$$m_corrected / \sqrt{v_corrected + \varepsilon},$$

где *m_corrected* и *v_corrected* вычисляется исходя из обновленных моментов.

Шаг 7. Полное обновление весов первого слоя (Adam).

Выполнить обновление всех весов первого слоя:

$$w^{(0)} = w^{(0)} - \alpha \cdot w_0_update.$$

Шаг 8. Выбор веса для обновления в третьем слое.

Вычислить $cw_2_update = |w_2_update| \odot mask_w_2$.

Найти индекс p максимального элемента в cw_2_update. Создать $w^{*(2)} = w^{(2)}$ и обновить выбранный вес:

$$w^{*(2)}{}_p = w^{*(2)}{}_p - \beta \cdot w_2_update_p.$$

Вычислить J_{new}.

Шаг 9. Проверка улучшения для третьего слоя.

Если $J_{new} < J$: принять новое значение $w^{(2)}{}_p = w^{*(2)}{}_p$, обновить $J = J_{new}$.

Иначе: деактивировать вес в маске $mask_{индекс} = 0$ (*индекс* – номер p-го параметра в векторе *mask*).

Шаг 10. Выбор веса и обновление для второго слоя.

Найти индексы b, v максимального элемента в $cw_1_update = |w_1_update| \odot mask_w_1$.

Создать $w^{*(1)}$ и обновить выбранный вес:

$$w^{*(1)}_{b,v} = w^{*(1)}_{b,v} - \beta \cdot w_1_update_{i,j}.$$

Вычислить J_{new}.

Если $J_{new} < J$: принять новое значение $w^{(1)}_{b,v} = w^{*(1)}_{b,v}$, обновить $J = J_{new}$.

Иначе: деактивировать вес в маске $mask_{индекс} = 0$. Увеличить число итераций k, перейти к шагу 2.

В алгоритме Adam-S1 для обучения второго и последующих слоев используется алгоритм, представленный в п.3.2.1.

3.4.2. Результаты экспериментов

Параметры настройки стандартного алгоритма Adam, а также разработанных гибридных алгоритмов Adam-S-1 и Adam-S-2, использованные в ходе экспериментов, приведены в таблицах 3.13 и 3.14 соответственно. Для настройки гиперпараметров применялась процедура кросс-валидации на обучающей выборке. Поскольку гибридные алгоритмы сочетают полное обновление первого слоя методом Adam и разреживающее обновление последующих слоев, оптимизации подлежат три параметра: скорость обучения α для Adam, шаг изменения весов β для разреживающего механизма, а также максимальное число итераций *iterations*. Оптимальная комбинация гиперпараметров выбиралась по минимуму средней ошибки на валидационных фолдах, после чего проводилось финальное обучение на полной обучающей выборке с выбранными значениями параметров.

Таблица 3.13

Архитектурные гиперпараметры

Гиперпараметр	Значение
hidden_neurons	[8, 16, 32, 64]
f_1	[sigmoid, linear, relu, cloglogm]

Таблица 3.14

Гиперпараметры методов

Метод	Гиперпараметр	Значение
Adam	α	[0,01, 0,001, 0,0001]
	число эпох	[100, 300, 500, 700, 1000]
	μ_1	0,9
	μ_2	0,99
	ε	10^{-8}
Adam-S-1	α	[0,01, 0,001, 0,0001]
	β	[0,0001, 0,001, 0,009, 0,1]
	T	[100, 300, 500, 700, 1000]
	μ_1	0,9
	μ_2	0,99
	ε	10^{-8}
Adam-S-2	α	[0,01, 0,001, 0,0001]
	β	[0,0001, 0,001, 0,009, 0,1]
	T	[100, 300, 500, 700, 1000]
	μ_1	0,9
	μ_2	0,99
	ε	10^{-8}

Экспериментальная оценка разработанных алгоритмов проводилась в рамках трех вычислительных сценариев. Первый сценарий предусматривал прогнозирование цены закрытия акции на основе 30 технических индикаторов и пяти лаговых переменных. Технические индикаторы включали следующие показатели: простую и экспоненциальную скользящие средние, индекс направленного движения, Арун, индикатор тренда, канал Дончяна, индекс товарного канала, индекс денежного потока, балансовый объем, схождение/расхождение скользящих средних, индекс относительной силы, стохастический осциллятор, осциллятор Чанде, индикатор «Знать наверняка», тройную экспоненциальную скользящую среднюю, индикатор скорости изменения, полосы Боллинджера и ценовые полосы. В качестве объекта исследования выбрана акция ПАО «Артген» (ABIO), для которой использованы данные за последние 1000 торговых дней по состоянию на 11 февраля 2026 года.

Для оценки эффективности рассматриваемых алгоритмов обучения использовались MSE и MAE, вычисленные на обучающей

(train) и тестовой (test) выборках, а также показатель разреженности (Sparsity).

Перед обучением выполнялось нормирование выходной величины с использованием MinMax-преобразования.

Результаты моделирования представлены в таблице 3.15, где MSE CV обозначает среднюю квадратичную ошибку, полученную в ходе 5-кратной кросс-валидации. Гибридный алгоритм Adam-S-2 показал наилучшую обобщающую способность, достигнув минимальных значений MSE CV (0,000122–0,000129) при всех архитектурах. Также Adam-S-2 обеспечил стабильно низкие значения тестовой MSE (0,000186–0,000196) и MAE (0,0107–0,0109) при умеренной разреженности (19–73%). Алгоритм Adam-S-1 продемонстрировал наиболее высокую разреженность, достигающую 91% при 32 нейронах, однако ценой существенного роста ошибки на тестовой выборке (MSE test = 0,002292, MAE test = 0,0388 для 32 нейронов). Стандартный Adam, несмотря на отсутствие разреженности, показал устойчивые результаты с тестовой MSE в диапазоне 0,000239–0,000301, однако уступил Adam-S-2 по точности на всех конфигурациях. Полученные результаты свидетельствуют о том, что предложенный гибридный подход Adam-S-2 позволил достичь оптимального баланса между точностью прогнозирования и разреженностью модели независимо от числа нейронов в скрытых слоях.

Таблица 3.15

Результаты моделирования цены акции

Число нейронов	Метрика	Adam	Adam-S-1	Adam-S-2
8	MSE CV	0,000531	0,003127	0,000129
	MSE train	0,000169	**0,000049**	0,000085
	MAE train	0,00941	**0,004931**	0,006274
	MSE test	0,000301	**0,000091**	0,000186
	MAE test	0,014084	**0,007194**	0,010709
	Число ненулевых параметров	369	272	299
	Sparsity	0	0,26	0,19

Окончание таблицы 3.15

Число нейронов	Метрика	Adam	Adam-S-1	Adam-S-2
16	MSE CV	0,000465	0,000435	**0,000122**
	MSE train	0,000161	0,00011	0,000087
	MAE train	0,009202	0,007524	0,006311
	MSE test	0,000277	0,000197	0,00019
	MAE test	0,013415	0,011446	0,010851
	Число ненулевых параметров	865	379	595
	Sparsity	0	0,56	0,31
32	MSECV	0,001134	0,002517	0,000124
	MSE train	0,000096	0,000409	0,000094
	MAE train	0,007129	0,014881	0,006755
	MSE test	0,000239	0,002292	0,000196
	MAE test	0,012177	0,038848	0,011083
	Число ненулевых параметров	2241	192	854
	Sparsity	0	0,91	0,62
64	MSECV	0,000576	0,000301	0,000123
	MSE train	0,000096	0,000058	0,000086
	MAE train	0,007305	0,00538	0,006316
	MSE test	0,00024	0,000187	0,000187
	MAE test	0,012492	0,010561	0,010732
	Число ненулевых параметров	6529	785	1779
	Sparsity	0	0,88	0,73

Второй сценарий экспериментального исследования был посвящен решению задачи прогнозирования рентабельности активов предприятий, рассмотренной в работе [36]. Временной горизонт исследования охватывал 2017–2020 гг. и разделен на обучающий (2017–2019) и тестовый (2020) периоды. Обучающая выборка представлена панельными данными и включает 1653 наблюдения (551 компания × 3 года), тестовая выборка содержит 551 наблюдение (551 компания × 1 год). Для каждого предприятия анализировались следующие показатели: посещаемость веб-сайта компании, возраст компании, рентабельность активов предыдущего года, рост продаж, доля основных средств в активах, коэффициент текущей лик-

видности, леверидж, оборачиваемость активов, а также переменная, отражающая отраслевые и временные различия в показателях деятельности фирм.

Рассмотрена архитектура, предложенная в работе [36], содержащая два скрытых слоя по 64 нейрона в каждом. Входные данные были нормированы с использованием z-нормализации, нормирование выходной величины не выполнялось. Результаты, представленные в таблице 3.16, свидетельствуют о том, что алгоритм Adam-S-2 обеспечил сопоставимую точность прогнозирования при существенной разреженности сети, достигшей 84%.

Таблица 3.16

Результаты моделирования финансовых показателей

Метрика	Adam	Adam-S-1	Adam-S-2
MSE train	51,17	130,28	56,55
MAE train	4,51	8,243	4,56
MSE test	73,49	179,46	73,3
MAE test	5,64	9,34	5,59
Sparsity	0	0,98	0,84
Число ненулевых параметров	4929	89	771

На рисунке 3.11 представлена диаграмма размаха квадратичной ошибки на тестовой выборке. Гибридный алгоритм Adam-S-2 характеризуется наименьшей медианой и наименьшим разбросом значений ошибки. В противоположность этому, алгоритм Adam-S-1 демонстрирует наибольшую медиану и максимальный разброс значений. Статистическая значимость различий между всеми методами подтверждена критерием Вилкоксона для парных выборок (p value < 0,001).

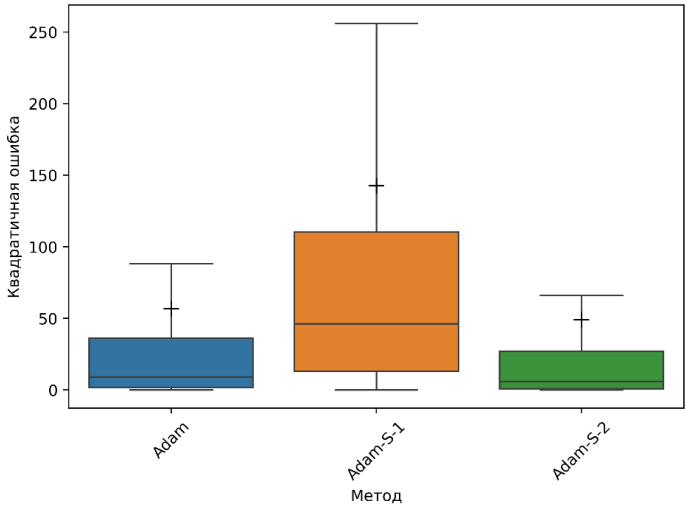


Рис. 3.11. Диаграмма размаха квадратичной ошибки

Третий сценарий вычислительных экспериментов предусматривал тестирование разработанных алгоритмов на стандартных наборах данных из репозитория UCI Machine Learning Repository. Для оценки базовой производительности моделей использован набор данных Auto MPG, содержащий 392 наблюдений и 8 признаков, с архитектурой сети из 8 нейронов в скрытом слое. Исследование способности алгоритмов моделировать нелинейные зависимости выполнялось на наборе данных Yacht Hydrodynamics (308 наблюдений, 6 признаков) с архитектурой из 16 нейронов. Для анализа склонности моделей к переобучению применялся набор данных Residential Building (372 наблюдений, 103 признаков) с архитектурой из 32 нейронов, что позволило оценить устойчивость алгоритмов при увеличении числа параметров модели. Входные и выходные данные были нормированы с использованием z-нормализации.

Анализ полученных результатов (таблица 3.16) показал, что на всех трех наборах данных гибридный алгоритм Adam-S-2 либо превосходил стандартный Adam, либо показывал сопоставимые результаты при значительно меньшем числе параметров. На наборе данных Auto MPG алгоритм Adam-S-2 продемонстрировал наилучшие показатели по всем метрикам, обеспечив снижение тестовой

MSE на 32% по сравнению со стандартным Adam (6,93 против 10,25) при разреженности 23%. Для нелинейной задачи Yacht Hydrodynamics Adam-S-2 также показал превосходство над базовым алгоритмом, уменьшив тестовую MSE с 0,561 до 0,395, что соответствует улучшению на 30% при разреженности 17%. На наборе данных Residential Building, предназначенном для оценки склонности моделей к переобучению, алгоритм Adam-S-2 продемонстрировал наилучшую устойчивость, достигнув тестовой MSE 535,03, что на 27% лучше Adam (732,30) и значительно превзошел Adam-S-1 (3965,97). При этом разреженность модели составила 40%, а число ненулевых параметров сократилось с 4513 до 2702. Алгоритм Adam-S-1, также как и в предыдущих сценариях, показал максимальную разреженность сети во всех трех задачах.

Таблица 3.16

Результаты моделирования
с использованием стандартных датасетов

Датасет	Метрика	Adam	Adam-S-1	Adam-S-2
Auto MPG	MSE CV	7,86	7,75	7,24
	MSE train	7,33	7,06	5,7
	MAE train	1,96	1,95	1,75
	MSE test	10,25	10,23	6,93
	MAE test	2,38	2,43	1,92
	Sparsity	0	0,43	0,23
	Число ненулевых параметров	145	82	112
Yacht Hydrodynamics	MSE CV	0,67	3,78	0,65
	MSE train	0,3	2,2	0,09
	MAE train	0,42	1,22	0,20
	MSE test	0,56	2,86	0,39
	MAE test	0,51	1,27	0,31
	Sparsity	0	0,7	0,17
	Число ненулевых параметров	401	120	333
Residential Building	MSE CV	575,85	6111,1	531,27
	MSE train	24,28	7181	45,59
	MAE train	3,78	50,36	3,83
	MSE test	732,3	3965,97	535,03
	MAE test	16,03	43,13	17,02
	Sparsity	0	0,71	0,4
	Число ненулевых параметров	4513	1297	2702

Анализ распределения наилучших активационных функций в исследуемых сценариях (рисунок 3.12) показал, что для Adam в большинстве задач оптимальной являлась relu, для Adam-S-1 – cloglogm, а для Adam-S-2 – линейная функция. При этом наилучшая функция активации зависит от специфики задачи: так, например, на нелинейном датасете Yacht Hydrodynamics все три метода продемонстрировали наивысшую эффективность с функцией cloglogm.

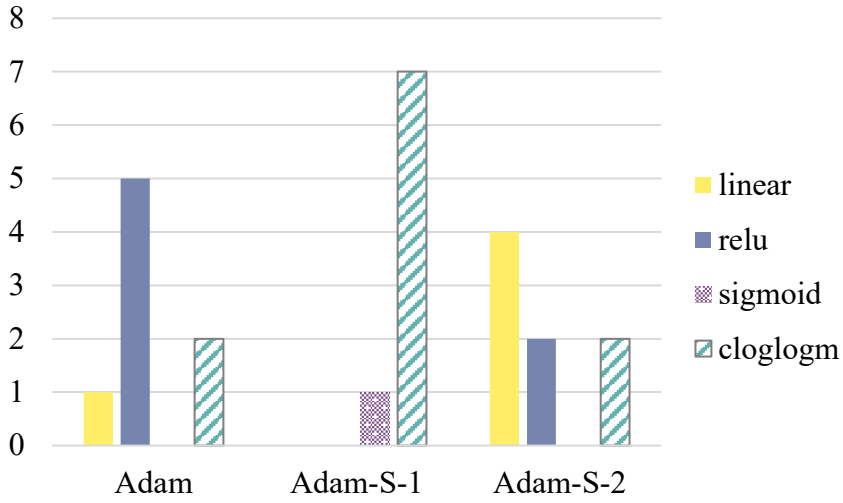


Рис. 3.12. Распределение активационных функций

ЗАКЛЮЧЕНИЕ

В монографии приведены алгоритмы обучения регрессионных и нейросетевых моделей, а также кластеризации данных на основе методов решения обратных задач. Обобщая полученные результаты, можно сделать следующие основные выводы:

1. Алгоритмы на основе стохастического метода решения обратных задач (рассмотренные во второй главе) показали работоспособность при решении задач глобальной оптимизации, обучения полносвязных нейронных сетей и кластеризации данных. Адаптивный стохастический поиск позволил в ряде случаев повысить точность предсказаний (особенно алгоритм показал эффективность для временных рядов и специализированных функций активаций, таких как sincos) даже при минимальной настройке гиперпараметров, что делает предложенные алгоритмы привлекательными в качестве решений «из коробки».

2. Методы решения обратных задач с L1-регуляризацией (третья глава) показали возможность создания компактных и интерпретируемых моделей. В частности, адаптация этого подхода для обучения LASSO-регрессии и нейронных сетей подтвердила его способность формировать сильно разреженные решения и отсеять незначимые признаки.

Стоит отметить, что эксперименты были проведены преимущественно на сетях с одним-двумя скрытыми слоями или однослойных, однако алгоритмы могут быть расширены и для более глубокий сетей.

Предложенные алгоритмы и результаты могут быть полезны для разработки широкого спектра приложений в различных сферах науки, экономики и производства. Материалы данной монографии могут быть использованы в дальнейших междисциплинарных исследованиях и предоставляют специалистам инструментарий для повышения качества, устойчивости и интерпретируемости современных моделей машинного обучения.

ЛИТЕРАТУРА

1. Тимофеев, Ю. М. Математические аспекты решения обратных задач атмосферной оптики / Ю. М. Тимофеев, А. В. Поляков. – Санкт-Петербург: Изд-во С.-Петерб. ун-та, 2001. – 188 с.

2. Ватульян, А. О. Обратные задачи в механике деформируемого твердого тела / А. О. Ватульян. – Москва: Физматлит, 2007. – 223 с.

3. Тихонов, А. Н. Об устойчивости обратных задач / А. Н. Тихонов // Докл. АН СССР. – 1943. – № 5 (39). – С. 195–198.

4. Тихонов, А. Н. Методы решения некорректно поставленных задач / А. Н. Тихонов, В. Я. Арсенин. – Москва: Наука, 1974. – 223 с.

5. Одинцов, Б. Е. Обратные вычисления в формировании экономических решений / Б. Е. Одинцов. – Москва: Финансы и статистика, 2004. – 192 с.

6. Грибанова, Е. Б. Стохастические алгоритмы решения обратных задач экономического анализа с ограничениями / Е. Б. Грибанова // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2016. – № 4. – С. 112–116.

7. Грибанова, Е. Б. Разработка стохастического алгоритма для решения обратных иерархических задач прикладной экономики / Е. Б. Грибанова // Информационные системы и технологии в образовании, науке и бизнесе : сб. материалов всероссийской научно-практической конференции, Улан-Удэ. – Улан-Удэ: БГУ, 2022. – С. 23–28.

8. Zhenga, G.-H. Solving the backward problem for space-fractional diffusion equation by a fractional Tikhonov regularization method / G.-H. Zhenga, Q.-G. Zhang // Mathematics and Computers in Simulation. – 2018. – № 148. – P. 37–47.

9. Park, Y. Parameter determination for Tikhonov regularization problems in general form / Y. Park, L. Reichel, G. Rodriguez, X. Yud // Journal of Computational and Applied Mathematics. – 2018. – № 343. – P. 12–25.

10. Modulus-based iterative methods for constrained Tikhonov regularization / Z.-Z. Bai, A. Buccini, K. Hayami [et al.] // Journal of Computational and Applied Mathematics. – 2017. – № 319. – P. 1–13.

11. Wang, H. Cauchy sparse NMF with manifold regularization: A robust method for hyperspectral unmixing / H. Wang, W. Yang, N. Guan // Knowledge-Based Systems. – 2019. – № 184. – P. 1–16.

12. Scardapane, S. Group sparse regularization for deep neural networks / S. Scardapane, D. Comminiello, A. Hussain, A. Uncini // Neurocomputing. – 2017. – № 241. – P. 81–89.

13. Денисов, А. М. Введение в теорию обратных задач : учеб. пособие / А. М. Денисов. – Москва: Изд-во МГУ, 1994. – 207 с.

14. Boyd, S. Convex Optimization / S. Boyd, L. Vandenberghe. – Cambridge: Cambridge University Press, 2004. – 730 p.

15. Воскобойников, Ю. Е. Выбор параметра регуляризации при решении обратных измерительных задач / Ю. Е. Воскобойников, Н.Г. Преображенский // Автометрия. – 1984. – № 2. – С. 31–38.

16. Воскобойников, Ю. Е. Выбор параметра регуляризации и ошибки восстановления входного сигнала в методе статистической регуляризации / Ю. Е. Воскобойников // Автометрия. – 1975. – № 4. – С. 10–18.

17. Assessment of Tikhonov-type regularization methods for solving atmospheric inverse problems / J. Xu, F. Schreier, A. Doicu, T. Trautmann // Journal of Quantitative Spectroscopy & Radiative Transfer. – 2016. – № 184. – P. 274–286.

18. Gribanova, E. B. Algorithm for solving the inverse problems of economic analysis in the presence of limitations / E. B. Gribanova // EUREKA: Physics and Engineering. – 2020. – № 1. – P. 70–78.

19. Gribanova, E. Elaboration of an Algorithm for Solving Hierarchical Inverse Problems in Applied Economics / E. Gribanova // Mathematics. – 2022. – № 10 (15).

20. Ганичева, А. В. Метод решения некоторых классов оптимизационных задач / А. В. Ганичева // Моделирование, оптимизация и информационные технологии. – 2019. – № 2 (7). – С. 43–54.

21. Wu, L. A neural network transformation based global optimization algorithm / L. Wu, H. Chen, Z. Yang // Information Sciences. – 2025. – V. 694. – 121693. – DOI 10.1016/j.ins.2024.121693.

22. Bottou, L. Optimization methods for large-scale machine learning / L. Bottou, F.E. Curtis, G. Nocedal // SIAM Review. – 2018. – V. 60, № 2. – P. 223–311.

23. Decision-Focused Learning: Foundations, State of the Art, Benchmark and Future Opportunities / J. Mandi, J. Kotary, S. Berden [et al.] // Journal of Artificial Intelligence Research. – 2024. – V. 80. – P. 1623–1701.

24. Ghazaan, M. A novel adaptive optimization scheme for advancing metaheuristics and global optimization / M. Ghazaan, A. Oshnari, A. Oshnari // Swarm and Evolutionary Computation. – 2024. – V. 91. – 101779. – DOI 10.1016/j.swevo.2024.101779.

25. Wolpert, D. H. No Free Lunch Theorems for Optimization / D. H. Wolpert, W. G. Macready // IEEE Transactions on Evolutionary Computation. – 1997. – V. 1, № 1. – P. 67–82. – DOI 10.1109/4235.585893.

26. Bumin, M. Predicting the direction of financial dollarization movement with genetic algorithm and machine learning algorithms: The case of Turkey / M. Bumin, M. Ozcalici // Expert Systems with Applications. – 2023. – V. 213. – 119301. – DOI 10.1016/j.eswa.2022.119301.

27. An artificial bee colony algorithm with adaptive heterogeneous competition for global optimization problems / X. Chu, F. Cai, J. Cui [et al.] // Applied Soft Computing. – 2020. – V. 93. – 106391. – DOI 10.1016/j.asoc.2020.106391.

28. Accelerated grey wolf optimization for global optimization problems / R. Rajakumar, K. Sekaran, C.-H. Hsu, S. Kadry // Technological Forecasting and Social Change. – 2021. – V. 169. – 120824. – DOI 10.1016/j.techfore.2021.120824.

29. The arithmetic optimization algorithm / L. Abualigah, A. Diabat, S. Mirjalili [et al.] // Computer Methods in Applied Mechanics and Engineering. – 2021. – V. 376. – 113609. – DOI 10.1016/j.cma.2020.113609.

30. Ziying, Y. Group Teaching Optimization Algorithm: A novel metaheuristic method for solving global optimization problems / Y. Ziying, J. Zhigang // Expert Systems with Applications. – 2020. – V. 148. – 113246. – DOI 10.1016/j.eswa.2020.113246.

31. Das, B. Student psychology-based optimization algorithm: A new population-based optimization algorithm for solving optimization problems / B. Das, V. Mukherjee, D. Das // Advances in Engineering Software. – 2020. – V. 146. – 102804. – DOI 10.1016/j.advengsoft.2020.102804.

32. Gribanova, E. An algorithm for solving inverse problems characterized by multi-extremal functions / E. Gribanova // International Journal on Information Technologies and Security. – 2025. – V. 17, № 3. – P. 13–24. – DOI 10.59035/caxd7851.

33. Gribanova, E. An Inverse-Based Algorithm for Global Optimization / E. Gribanova, D. Leonov // Journal of Sustainable Development Innovations. – 2026. – V. 3, № 1. – P. 52–61. – DOI 10.61552/JSI.2026.01.006.

34. Plevris, V. A Collection of 30 Multidimensional Functions for Global Optimization Benchmarking / V. Plevris, G. Solorzano // Data. – 2022. – V. 7, № 4. – 46. – DOI 10.3390/data7040046.

35. Wallner, M. Predictors of average wait time at an intersection using artificial neural network / M. Wallner, T. Peráček // International Journal for Quality Research. – 2023. – V. 17, № 3. – P. 867–890. – DOI 10.24874/IJQR17.03-15.

36. Predicting the performance of retail market firms: regression and machine learning methods / D.B. Vukovic, L. Spitsina, E. Gribanova [et al.] // Mathematics. – 2023. – V. 11, № 8. – 1916. – DOI 10.3390/math11081916.

37. A Unified Analysis of AdaGrad With Weighted Aggregation and Momentum Acceleration / L. Shen, C. Chen, F. Zou [et al.] // IEEE Transactions on Neural Networks and Learning Systems. – 2023. – V. 35. – P. 14482–14490. – DOI 10.1109/TNNLS.2023.3279381.

38. Comparison of sgd, adadelta, adam optimization in gender classification using cnn / S. Mandasari, D. Irfan, Wanayumini, R. Rosnelly // JURTEKSI (Jurnal Teknologi dan Sistem Informasi). – 2023. – V. 9, № 3. – P. 345–354. – DOI 10.33330/jurteksi.v9i3.2067.

39. Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning / R. Elshamy, O. Abu Elnasr, M. Elhoseny, S. Elmougy // Scientific Reports. – 2023. – V. 13. – 8814. – DOI 10.1038/s41598-023-35663-x.

40. Ding, S. An optimizing BP neural network algorithm based on genetic algorithm / S. Ding, C. Su, J. Yu // Artificial Intelligence Review. – 2011. – V. 36. – P. 153–162. – DOI 10.1007/s10462-011-9208-z.

41. Applying artificial neural networks (ANNs) to solve solid waste-related issues: A critical review / A. Xu, H. Chang, Y. Xu [et al.] // Waste Management. – 2021. – V. 124. – P. 385–402. – DOI 10.1016/j.wasman.2021.02.029.

42. Gribanova, E. Application of inverse optimization algorithms in neural network models for short-term stock price forecasting / E. Gribanova, R. Gerasimov, E. Viktorenko // Big Data and Cognitive Computing. – 2025. – V. 9, № 9. – 235. – DOI 10.3390/bdcc9090235.

43. Gomes, G. S. Comparison of new activation functions in neural network for forecasting financial time series / G. S. Gomes, T. B. Ludermir, L.M. Lima // Neural Computing and Applications. – 2011. – V. 20. – P. 417–439. – DOI 10.1007/s00521-010-0407-3.

44. Ziyin, L. Neural networks fail to learn periodic functions and how to fix it / L. Ziyin, T. Hartwig, M. Ueda // Advances in Neural Information Processing Systems 33 (NeurIPS 2020). – Vancouver, Canada, 2020. – P. 1–12.

45. Booth, A. Geographic scale in air quality clustering: a dynamic time warping approach for smarter city policies / A. Booth, P. James, S. McGough // City and Environment Interactions. – 2026. – V. 30. – 100307. – DOI 10.1016/j.cacint.2026.100307.

46. Use of clustering techniques for clinical and epidemiological research: practical tips using an example from rheumatology / M. Jansen, M.M. Bakker, A.R. Sepriano [et al.] // Journal of Clinical Epidemiology. – 2026. – V. 192. – 112183. – DOI 10.1016/j.jclinepi.2026.112183.

47. Tang, Y. Fuzzy C-Means clustering through SSIM and patch for image segmentation / Y. Tang, F. Ren, W. Pedrycz // Applied Soft Computing. – 2020. – V. 87. – 105928. – DOI 10.1016/j.asoc.2019.105928.

48. Zainuddin, Z. Balanced clustering for student admission school zoning by parameter tuning of constrained k-means / Z. Zainuddin, A.A.N. Risal // IAES International Journal of Artificial Intelligence. – 2024. – V. 13. – P. 2301–2313.

49. Jain, A. K. Data clustering: 50 years beyond K-Means / A. K. Jain // Pattern Recognition Letters. – 2010. – V. 31, № 8. – P. 651–666. – DOI 10.1016/j.patrec.2009.09.011.

50. Top 10 algorithms in data mining / X. Wu, V. Kumar, J.R. Quinlan [et al.] // Knowledge and Information Systems. – 2007. – V. 14, № 1. – P. 1–37. – DOI 10.1007/s10115-007-0114-2.

51. Model-based clustering of music pieces / Y. Zhang, X. Zhu, H. Tong, V. Melnykov // Pattern Recognition. – 2026. – V. 178. – 113362. – DOI 10.1016/j.patcog.2026.113362.

52. Hashemi, S.E. A fuzzy C-means algorithm for optimizing data clustering / S.E. Hashemi, F. Gholian-Jouybari, M. Hajiaghaei-Keshteli // Expert Systems with Applications. – 2023. – V. 227. – 120377. – DOI 10.1016/j.eswa.2023.120377.

53. Hashemi, Z. Initializing K-means clustering algorithm based on frequent patterns / Z. Hashemi, M. Amiri, M. Naderi // Applied Soft Computing. – 2026. – V. 193. – 114838. – DOI 10.1016/j.asoc.2026.114838.

54. Maeyer, R. Balanced k-means revisited / R. Maeyer, S. Sieranoja, P. Franti // Applied Computational Intelligence. – 2023. – V. 3. – P. 145–179. – DOI 10.3934/aci.2023008.

55. Population-based clustering of co-occurring social determinants: an application of unsupervised machine learning / I. Giesinger, E. Buajitti, A. Siddiqi [et al.] // Annals of Epidemiology. – 2026. – V. 117. – 110062. – DOI 10.1016/j.annepidem.2026.110062.

56. MacQueen, J. B. Some methods for classification and analysis of multivariate observations / J. B. MacQueen // Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. – Berkeley, USA, 1967. – P. 281–297.

57. Bezdek, J. C. FCM: The fuzzy c-means clustering algorithm / J. C. Bezdek, R. Ehrlich, W. Full // Computers & Geosciences. – 1984. – V. 10, № 2–3. – P. 191–203. – DOI 10.1016/0098-3004(84)90020-7.

58. Jouan, G. Gaussian mixture models for clustering and calibration of ensemble weather forecasts / G. Jouan, A. Cuzol, V. Monbet, G. Monnier // Discrete and Continuous Dynamical Systems – Series S. – 2023. – V. 16, № 2. – P. 309–328. – DOI 10.3934/dcdss.2022037.

59. Nocedal, J. Numerical Optimization / J. Nocedal, S. J. Wright. – New York: Springer, 2006. – 664 p.

60. Storn, R. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces / R. Storn, K. Price // Journal of Global Optimization. – 1997. – V. 11, № 4. – P. 341–359. – DOI 10.1023/A:1008202821328.

61. Scikit-learn: Machine learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort [et al.] // Journal of Machine Learning Research. – 2011. – V. 12. – P. 2825–2830.

62. SciPy 1.0: Fundamental algorithms for scientific computing in Python / P. Virtanen, R. Gommers, T.E. Oliphant [et al.] // Nature Methods. – 2020. – V. 17. – P. 261–272. – DOI 10.1038/s41592-019-0686-2.

63. UCI Machine Learning Repository. – URL: http://archive.ics.uci.edu/ml (дата обращения: 31.05.2026).

64. SDC: A parameter-free clustering algorithm for incomplete datasets / Q. Li, X. Zeng, S. Wang [et al.] // Pattern Recognition. – 2026. – V. 177. – 113363. – DOI 10.1016/j.patcog.2026.113363.

65. Laakel Hemdanou A. Cover Tree-Optimized Spectral Clustering: Efficient Nearest Neighbor Search for Large-Scale Data Partitioning / A. Laakel Hemdanou, Y. Achtoun, S. Mouali, M. Lamarti Sefian, V. Šešum Čavić, S. Radenović // Machine Learning and Knowledge Extraction. – 2025. – V. 7. – 139. – DOI 10.3390/make7040139.

66. Kirmizi, M. Performance and energy analysis of turboprop engine for air freighter aircraft with the aid of multiple regression / M. Kirmizi, H. Aygun, O. Turan // Energy. – 2023. – V. 283. – 129084. – DOI 10.1016/j.energy.2023.129084.

67. Xu, W. Analysis and simulation of the driving mechanism and ecological effects of land cover change in the Weihe River Basin, China / W. Xu, J. Song, Y. Long, R. Mao, B. Tang, B. Li // Journal of Environmental Management. – 2023. – V. 344. – 118320. – DOI 10.1016/j.jenvman.2023.118320.

68. Thwe, W. P. The regression models for lifelong learning competencies for teacher trainers / W. P. Thwe, A. Kálmán // Heliyon. – 2023. – V. 9. – e13749. – DOI 10.1016/j.heliyon.2023.e13749.

69. The impact of platform economy on enterprise value mediated by technological innovation / X. Pan, Y. Wang, C. Zhang [et al.] // Journal of Business Research. – 2023. – V. 165. – 114051. – DOI 10.1016/j.jbusres.2023.114051.

70. Gribanova, E. Forecasting Dynamics of Stock Based on Sparse Neural Networks and Large Language Models / E. Gribanova, E. Viktorenko, R. Gerasimov // Journal of Corporate Finance Research. – 2025. – V. 19, № 4. – P. 88–104. – DOI 10.17323/j.jcfr.2073-0438.19.4.2025.88-104.

71. Грибанова, Е. Б. Алгоритм выбора признаков линейной регрессии для решения проблемы мультиколлинеарности / Е. Б. Грибанова // Искусственный интеллект и принятие решений. – 2025. – № 1. – С. 46–55. – DOI 10.14357/20718594250104.

72. Gribanova, E. An Inverse-Based Algorithm for Solving Multicollinearity Problem / E. Gribanova // La Matematica. – 2026. – V. 5. – 41. – DOI 10.1007/s44007-026-00222-z.

73. Melkumova, L. E. Comparing ridge and lasso estimators for data analysis / L. E. Melkumova, S. Y. Shatskikh // Procedia Engineering. – 2017. – V. 201. – P. 746–755.

74. Interpreting lasso regression model by feature space matching analysis for spatiotemporal correlation based wind power forecasting / Y. Zhao, Y. Zhao, H. Liao [et al.] // Applied Energy. – 2025. – V. 380. – 124954. – DOI 10.1016/j.apenergy.2024.124954.

75. Roozbeh, M. Generalized cross validation for simultaneous optimization of tuning parameters in ridge regression / M. Roozbeh, M. Arashi, N.A. Hamzah // Iranian Journal of Science and Technology, Transactions A: Science. – 2020. – V. 44. – P. 473–485.

76. Грибанова, Е. Б. Гибридный алгоритм построения разреженной регрессии / Е. Б. Грибанова, Р. С. Герасимов // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2025. – Т. 28, № 1. – С. 86–92. – DOI 10.21293/1818-0442-2024-28-1-86-92.

77. Gribanova, E. Inverse-based Lasso Regression: Novel Algorithms for Feature Selection and Multicollinearity Mitigation / E. Gribanova, R. Gerasimov // Intelligenza Artificiale. – 2026. – DOI 10.1177/172480352614804.

78. Gribanova, E. Identifying key predictors of academic performance in the context of higher education digitalization: a regression analysis with regularized models / E. Gribanova, V. Petuhova // European Journal of Contemporary Education. – 2025. – V. 14, № 3. – P. 258–267. – DOI 10.13187/ejced.2025.3.258.

79. Gribanova, E. An algorithm for training neural networks with L1 regularization / E. Gribanova, R. Gerasimov // IAES International Journal of Artificial Intelligence. – 2025. – V. 14, № 5. – P. 3781–3789. – DOI 10.11591/ijai.v14.i5.pp3781-3789.

80. Kaveh, M. Application of meta-heuristic algorithms for training neural networks and deep learning architectures: a comprehensive review / M. Kaveh, M.S. Mesgari // Neural Processing Letters. – 2023. – V. 55, № 4. – P. 4519–4622. – DOI 10.1007/s11063-022-11055-6.

81. Boosting the training of neural networks through hybrid metaheuristics / M. A. Al-Betar, M. A. Awadallah, I. Abu Doush [et al.] // Cluster Computing. – 2023. – V. 26, № 3. – P. 1821–1843. – DOI 10.1007/s10586-022-03708-x.

82. Xue, Y. A hybrid training algorithm based on gradient descent and evolutionary computation / Y. Xue, Y. Tong, F. Neri // Applied Intelligence. – 2023. – V. 53, № 22. – P. 21465–21482. – DOI 10.1007/s10489-023-04595-4.

83. Mahapatra, A. K. Quantized Orthogonal Experimentation SSA (QOX-SSA): A Hybrid Technique for Feature Selection (FS) and Neural Network Training / A. K. Mahapatra, N. Panda, B. K. Pattanayak // Arabian Journal for Science and Engineering. – 2025. – V. 50, № 2. – P. 1025–1056. – DOI 10.1007/s13369-024-09113-3.

84. Грибанова, Е. Б. Гибридный подход к обучению нейронных сетей: комбинирование метода Adam и разреживающих алгоритмов / Е. Б. Грибанова // Научно-техническая информация. Серия 2: Информационные процессы и системы. – 2026. – № 5. – С. 1–10. – DOI 10.36535/0548-0027-2026-05-1.

Научное издание

Грибанова Екатерина Борисовна
Герасимов Роман Сергеевич

**АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ
НА ОСНОВЕ МЕТОДОВ РЕШЕНИЯ ОБРАТНЫХ ЗАДАЧ**

Монография

Чебоксары, 2026 г.

Компьютерная верстка *Е. В. Кузнецова*

Подписано в печать 19.10.2026 г.
Дата выхода издания в свет 26.10.2026 г.
Формат 60×84/16. Бумага офсетная. Печать офсетная.
Гарнитура Times. Усл. печ. л. 5,81. Заказ К-1490. Тираж 500 экз.

Издательский дом «Среда»
428023, Чебоксары, Гражданская, 75, офис 12

+7 (8352) 655-731
info@phsreda.com
https://phsreda.com

Отпечатано в Студии печати «Максимум»
428023, Чебоксары, Гражданская, 75
info@maksimum21.ru
www.maksimum21.ru

ISBN 978-5-908172-30-1

9 785908 172301

Издательский дом «Среда»
Делитесь знаниями
в среде профессионалов!